AD-A095 735 ARMY ENGINEER WATERWAYS EXPERIMENT STATION VICKSBURG--ETC F/G 9/2
ANALYSIS OF AN ASSEMBLAGE OF DISCS EMPLOYING INTERACTIVE GRAPHI--ETC(U)
DEC 80 J B PALMERTON
UNCLASSIFIED WES/MP/GL-80-9 NL

1 OF 1
AD
A095735

END
DATE
FILMED
DTIC

LEVEL II

2

MISCELLANEOUS PAPER GL-80-9

ANALYSIS OF AN ASSEMBLAGE OF DISCS EMPLOYING INTERACTIVE GRAPHICS

by

John B. Palmerton

Geotechnical Laboratory
U. S. Army Engineer Waterways Experiment Station
P. O. Box 631, Vicksburg, Miss. 39180

December 1980

Final Report

Approved For Public Release; Distribution Unlimited

Prepared for Assistant Secretary of the Army (R&D)
Department of the Army
Washington, D. C. 20310

Under Project 4A161101A91D
Task 02, Work Unit 120

14 WES/MP/

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Miscellaneous Paper GL-80-9	2. GOVT ACCESSION NO. AD-A095135	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ANALYSIS OF AN ASSEMBLAGE OF DISCS EMPLOYING INTERACTIVE GRAPHICS		5. TYPE OF REPORT & PERIOD COVERED Final report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John B. Palmerton		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U. S. Army Engineer Waterways Experiment Station Geotechnical Laboratory P. O. Box 631, Vicksburg, Miss. 39180		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project 4A161101A91D Task 02, Work Unit 120
11. CONTROLLING OFFICE NAME AND ADDRESS Assistant Secretary of the Army (R&D) Department of the Army Washington, D. C. 20310		12. REPORT DATE December 1980 13. NUMBER OF PAGES 89
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer programs
DISC
Computer graphics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the development of a computer program entitled DISC. This computer program uses concepts of the distinct element method in that the kinematics of a system of particles (elements) are faithfully represented. The system of elements to be analyzed consists of a collection of individual elements with individual material properties rather than a continuum for which

(Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Continued).

material properties apply throughout the system. The types of particles that can be accommodated by DISC consist of discs and bar-shaped elements. These particles are permitted to react with one another by touching, rolling, bouncing, or sliding. Problems requiring a quasi-static solution or problems involving large velocities and displacement can be analyzed.

In addition to extending the distinct element method to disc-shaped particles, extensive use was made of interactive graphics. Through interactive graphics, the user of the program is relieved of the usual difficulties involved in data preparation and the plotting of results. The mode of operation of the computer program is visual; i.e., pictures (or plots) of the system of disc-shaped elements are drawn on a cathode ray terminal. The program operates in a time-sharing environment.

PREFACE

This study was authorized as part of the In-House Laboratory Independent Research (ILIR) Program for FY 78 and FY 79 and was performed under Project 4A161101A91D, Task 02, Work Unit 120, sponsored by the Assistant Secretary of the Army (R&D). The investigation was conducted during the period December 1977 to September 1980 at the U. S. Army Engineer Waterways Experiment Station (WES).

The study was conceived and conducted by Mr. J. B. Palmerton, Research Civil Engineer, Engineering Geology and Rock Mechanics Division (EGRMD), Geotechnical Laboratory (GL), under the general supervision of Dr. D. C. Banks, Chief, EGRMD, Mr. J. P. Sale, former Chief, GL, and Dr. P. F. Hadala, Assistant Chief, GL. This report was prepared by Mr. Palmerton.

Commanders and Directors of the WES during the conduct of this study were COL John L. Cannon, CE, and COL Nelson P. Conover, CE. Technical Director was Mr. Fred R. Brown.

CONTENTS

ANALYSIS OF AN ASSEMBLAGE OF DISCS EMPLOYING INTERACTIVE GRAPHICS

PART I: INTRODUCTION

Background

1. A computer modeling technique termed the "distinct" element method was introduced by Cundall (1971). Then, several years later, the technique was extended to analyze blocky rock systems (Cundall, 1974). This type of analysis permitted large-scale motions of the individual rock blocks. This work was further extended to the analysis of tunnel supports (Voegele, 1979). Thus, the distinct element method has been developed to accurately model many features of the behavior of jointed rock.

Purpose

2. This study was directed at developing methods for analyzing the behavior of systems of simple, particulate members. The basic shape of the particulate members was chosen to be a disc (or cylinder). The distinct element method is oriented more toward representing the kinematics of a system of particulate members instead of the continuum aspects. At the outset, the structure is viewed as a collection of individual members with individual properties, rather than a homogeneous system for which the material property effects are assumed to apply throughout.

3. In addition to extending the distinct element method to discs, there was a desire to use computer interactive graphics in the analysis so that the user could be relieved of laborious data preparation and time-consuming plotting of the results. This report describes the development of a computer program entitled DISC and contains discussions of the mathematical formulations, program organization, and program operation.

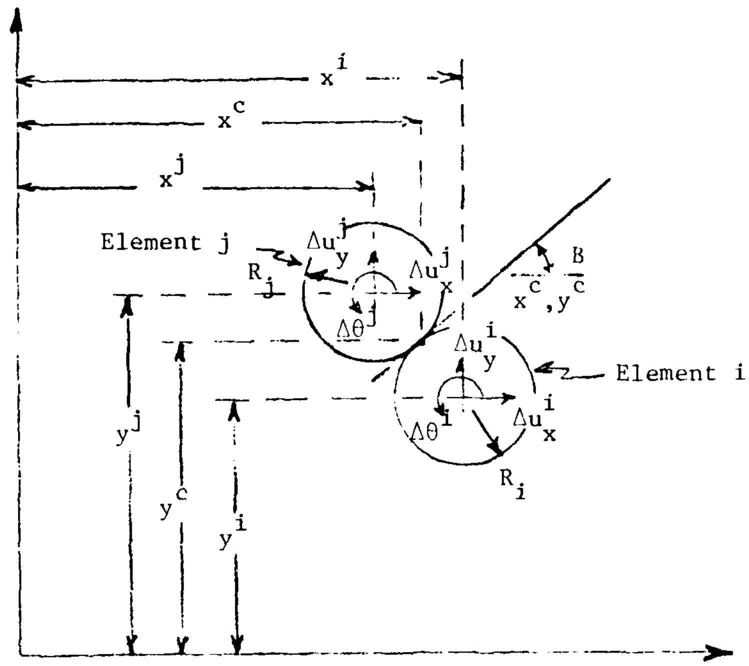
PART II: MATHEMATICAL CONCEPTS

4. The computer program DISC was formulated using concepts of the "distinct element" method proposed by Cundall (1974). Conceptually, the distinct element method computes the forces and displacements at points of contact between mathematically described particles (elements). The contact between the elements are mathematically represented by springs and dashpots. Any relative displacement between two elements in contact then results in a force developing within the springs and dashpots. The vector sum of all such forces on the discs in turn causes accelerations of the discs. If these accelerations are assumed to be constant for a prescribed interval of time, the associated velocities and displacements may be computed for the end of that time interval. These new displacements may then be used to determine new relative displacements between the contacting elements, and the cycle is then repeated over and over.

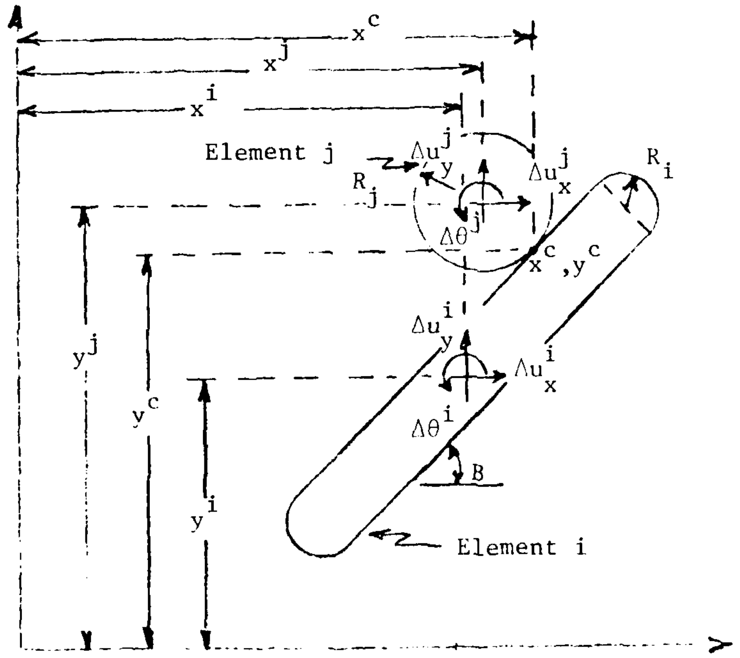
5. The computer program DISC is capable of handling element shapes corresponding to discs and bars. The bar elements are composed of two parallel sides of arbitrary length terminated on both ends by semicircles. Figure 1 shows two examples of two elements in contact. Condition a presents two disc elements in contact, and condition b a bar and a disc element in contact. Mathematically, the equations required to describe the subsequent motions of either contact pair are identical. However, condition b is somewhat more general and easier to visualize and as such will be referred to in the following discussion.

6. Figure 1b shows the bar element i, its centroid located at x^i, y^i, and the disc element j, its centroid located at x^j, y^j. The coordinates of the contact point are at x^c, y^c. The angle to the contact plane is denoted by B. (For the conditions shown in Figure 1a, B is the angle of the plane of tangency.)

7. The basic key equation of the distinct element method relates the incremental displacement of the contact point (into one of the two contacting bodies) to the relative motion of the elements that are in contact. The following equation is the result of superposition, i.e.,



a.



b.

Figure 1. Basic elements and definitions

first assume element i is fixed and determine the resulting motion of the contact, and then assume element j is fixed and examine the motion of the contact point. This analysis yields (as illustrated in Figure 2a)

$$\Delta u_y^c = \Delta u_y^j - \Delta u_y^i + \Delta\theta^j(x^c - x^j) - \Delta\theta^i(x^c - x^i) \tag{1}$$

and

$$\Delta u_x^c = \Delta u_x^j - \Delta u_x^i - \Delta\theta^j(y^c - y^j) + \Delta\theta^i(y^c - y^i)$$

where

Δu_y^c = the vertical component of the incremental contact displacement

Δu_x^c = the horizontal component of the incremental contact displacement

Δu_y^i = the current vertical incremental displacement of element i , etc. (i.e., the subscript (x or y) refers to the component direction and the superscript (i or j) refers to the element)

x^i, y^i = the centroid coordinates of element i

x^j, y^j = the centroid coordinates of element j

x^c = the horizontal coordinate of the contact point

y^c = the vertical coordinate of the contact point

$\Delta\theta$ = the incremental rotation of the element

8. The global angle of the contact plane is given by B . Thus, the horizontal and vertical relative contact displacements may be resolved into components parallel and perpendicular (shear and normal directions) to the contact plane. Thus, as shown in Figure 2b

$$\Delta u_s^c = \Delta u_y^c \sin B + \Delta u_x^c \cos B \tag{2}$$

and

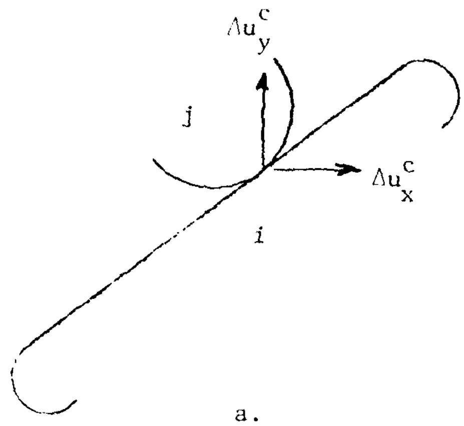
$$\Delta u_n^c = \Delta u_y^c \cos B - \Delta u_x^c \sin B$$

where

Δu_s^c = the component of the contact point displacement parallel to the contact plane

Δu_n^c = the normal component

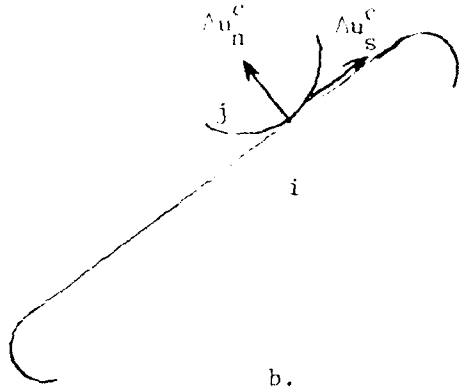
9. As previously mentioned, the contacts are represented by springs and dashpots (in both the normal and shear directions). Thus,



a.

$$\Delta u_y^c = \Delta u_y^j - \Delta u_y^i + \Delta\theta^j(x^c - x^j) - \Delta\theta^i(x^c - x^i)$$

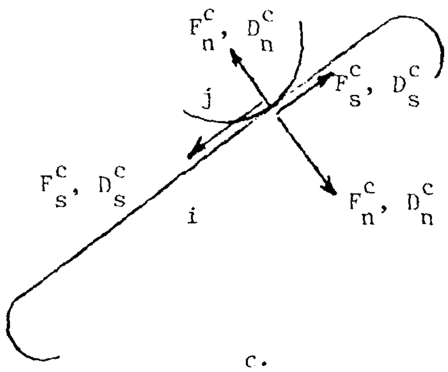
$$\Delta u_x^c = \Delta u_x^j - \Delta u_x^i - \Delta\theta^j(y^c - y^j) + \Delta\theta^i(y^c - y^i) \, .$$



b.

$$\Delta u_s^c = \Delta u_y^c \sin B + \Delta u_x^c \cos B$$

$$\Delta u_n^c = \Delta u_y^c \cos B - \Delta u_x^c \sin B$$



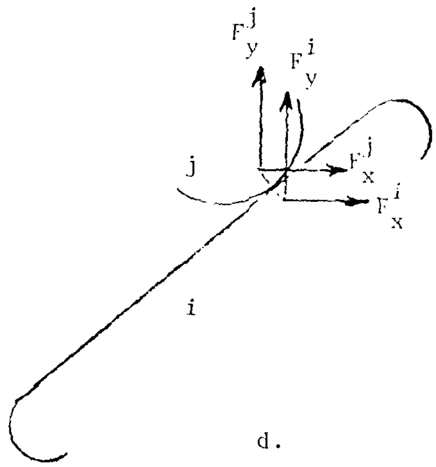
c.

$$F_n^c \doteq F_n^c - \Delta u_n^c k_n$$

$$F_s^c \doteq F_s^c + \Delta u_s^c k_s$$

$$D_n^c = - \Delta u_n^c K_n$$

$$D_s^c = \Delta u_s^c K_s$$



d.

$$F_y^i = (F_s^c + D_s^c) \sin B - (F_n^c + D_n^c) \cos B$$

$$F_x^j = (F_s^c + D_s^c) \cos B + (F_n^c + D_n^c) \sin B$$

$$F_y^i = - F_y^j$$

$$F_x^i = - F_x^j$$

Figure 2. Contact displacements and forces

the force generated within the springs and dashpots because of the incremental normal and shear displacements is given by

$$
\begin{aligned}
F_n^c &\doteq F_n^c - \Delta u_n^c k_n \\
F_s^c &\doteq F_s^c + \Delta u_s^c k_s \\
D_n^c &= -\Delta u_n^c K_n \\
D_s^c &= \Delta u_s^c K_s
\end{aligned}
\tag{3}
$$

where

F_n^c = the normal spring force

F_s^c = the shear spring force

D_n^c = the normal dashpot force

D_s^c = the shear dashpot force

k_n = the normai stiffness

k_s = the shear stiffness

K_n = the normal damping constant

K_s = the shear damping constant

The symbol $\doteq$ means "replaced by"; that is, after each increment of contact point displacement, new normal and shear forces are recomputed as functions of the old values. The dashpots are necessary to prevent the two contact elements from vibrating indefinitely. To be strictly correct, the dashpot forces should be related to the contact point velocity; however, for a time increment, the incremental displacement is proportional to the velocity (i.e., $\Delta u = v\Delta t$, where v is the relative velocity across the contact). Figure 2c shows the positive directions of the normal and shear forces. Also, note that a positive normal force indicates a compressive force.

10. Equation 3 is subject to modification if the tensile "strength" T (a negative value) is exceeded. Thus, it follows that if

$$F_n^c < T$$

then

$$F_n^c = F_s^c = D_n^c = D_s^c = 0$$

If F_n^c is less than T (which is normally set to zero), all forces at the contact are set to zero, i.e., the elements are tending to separate. In addition, if the coefficient of friction is given as μ, the equations are modified as follows:

If

$$\left| F_s^c \right| > \mu F_n^c$$

then

$$F_s^c = \mu F_n^c \left| F_s^c \right| / F_s^c$$

$$D_s^c = 0$$

Thus, the two elements will slide whenever the shear force exceeds the product of the coefficient of friction and the normal force, and the resulting shear force is maintained at the shear "strength."

11. Now that the contact shear and normal forces have been computed, all that remains is to resolve these forces back into the component directions. The result is

$$F_y^j = (F_s^c + D_s^c) \sin B - (F_n^c + D_n^c) \cos B$$

$$F_x^j = (F_s^c + D_s^c) \cos B + (F_n^c + D_n^c) \sin B \tag{4}$$

$$F_y^i = -F_y^j$$

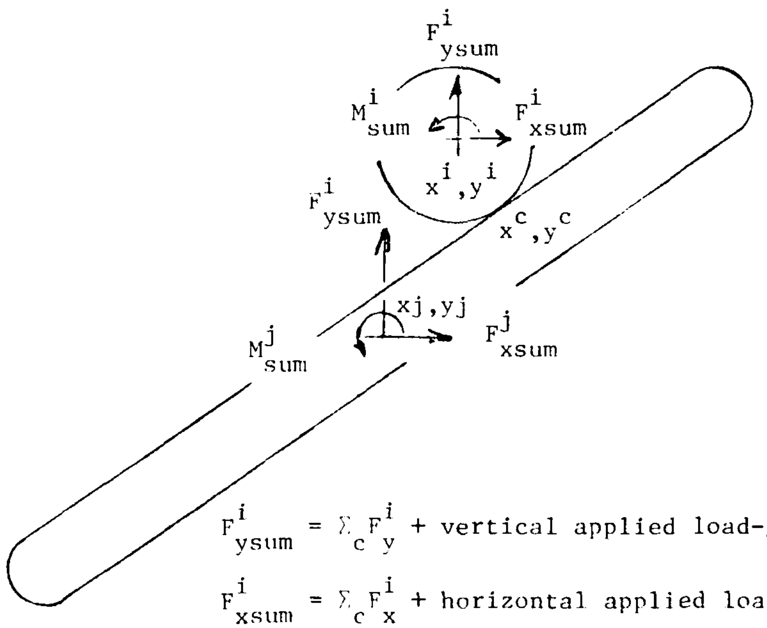
$$F_x^i = -F_x^j$$

where F_y^j is the vertical component of the contact force on element j, etc. Figure 2d shows the positive directions of the component forces. Obviously, the forces on element i are numerically equal but opposite to those on element j. Up to this point, only a single pair of contacting elements has been considered. In actuality, any element, say

element i, may possess a number of contacts. In addition, element i may be subject to applied forces and gravity forces. Thus, the total force acting on element i is given by

$$
\begin{aligned}
F^i_{ysum} &= \Sigma_c F^i_y + \text{vertical applied load} - \text{gravity load} \\
F^i_{xsum} &= \Sigma_c F^i_x + \text{horizontal applied load} \qquad (5) \\
M^i_{sum} &= \Sigma_c \left[F^i_y (x^c - x^i) - F^i_x (y^c - y^i) \right] + \text{applied moment}
\end{aligned}
$$

The symbol Σ_c means the summation for all contact points of element i. The equations are similar for element j. The term M^i_{sum} is the total moment acting on element i. Figure 3 illustrates the force sums.



$$
\begin{aligned}
F^i_{ysum} &= \Sigma_c F^i_y + \text{vertical applied load-gravity load} \\
F^i_{xsum} &= \Sigma_c F^i_x + \text{horizontal applied load} \\
M^i_{sum} &= \Sigma_c \left[F^i_y\,(x^c - x^i) - F^i_x\,(y^c - y^i) \right] \\
&\quad + \text{applied moment}
\end{aligned}
$$

Figure 3. Force sums on element i

12. After computing all the force sums acting on each element, the acceleration of each element may be computed from Newton's law of motion, i.e., acceleration = unbalanced force/mass, and angular acceleration = unbalanced moment/moment of inertia. Thus, the component accelerations a , and angular acceleration α are expressed as

$$
\begin{aligned}
a_x^i &= F_{xsum}^i/m^i \\
a_y^i &= F_{ysum}^i/m^i \\
\alpha^i &= M_{sum}^i/I^i
\end{aligned}
\tag{6}
$$

where

m^i = the mass of element i

I^i = the mass moment of inertia of element i

The velocities (ω is the angular velocity) are then computed by multiplying by the time step Δt :

$$
\begin{aligned}
v_x^i &\doteq v_x^i + a_x^i \Delta t \\
v_y^i &\doteq v_y^i + a_y^i \Delta t \\
\omega^i &\doteq \omega^i + \alpha^i \Delta t
\end{aligned}
\tag{7}
$$

A second numerical integration yields the incremental displacement for the element to be

$$
\begin{aligned}
\Delta u_x^i &= v_x^i \Delta t \\
\Delta u_y^i &= v_y^i \Delta t \\
\Delta \theta^i &= \omega^i \Delta t
\end{aligned}
\tag{8}
$$

The quantities above are sought in order to recycle the calculations. These quantities are introduced back into Equation 1, and the whole process is repeated.

13. Finally, the total displacement of the element centroids are found from

$$u_x^i \doteq u_x^i + \Delta u_x^i$$

$$u_y^i \doteq u_y^i + \Delta u_y^i \qquad (9)$$

$$\theta^i \doteq \theta^i + \Delta \theta^i$$

14. At the outset of each problem, the incremental displacements are, of course, zero. Thus, Equations 1 through 4 yield nothing, i.e., there are, up to now, no contact forces. In Equation 5, a force sum will result due to applied and/or gravity forces. Thus, an acceleration of an element will occur that in turn can yield incremental displacements. Equations 1 through 4 are then used to compute the new contact forces.

15. As noted from the discussion above, the mathematics involved in program DISC are quite simple, since only the concepts of a damped oscillator and Newton's law of motion are needed. The real difficulty in performing this type of analysis is the development of the logic to detect and to provide computer memory for the many contacts that may result. Thus, the problem becomes one of geometry and bookkeeping. The details of the program are presented in Part III.

PART III: PROGRAM ORGANIZATION

General Overview

16. The computer program DISC is a time-sharing FORTRAN program that consists of a main program and sixteen subroutines. Physically, DISC is composed of approximately 1200 FORTRAN statements. The program is written to use interactive graphics and is operated by a Tektronix 4014 (or 4010) terminal. With the exception of occasionally typing in a few numbers to redefine certain parameters, the commands primarily consist of single keystrokes. The output is primarily graphical, i.e., pictures of the present locations of the elements are drawn on the screen. While it is possible to obtain printed information, the normal mode of operation is visual.

17. As discussed in Part II, the actual calculations for determining the motion of the disc and bar elements are quite straightforward. Knowing the location of a contact point and the previous relative displacements of those elements in contact, the forces between the elements can be computed and then integrated over time to obtain new displacements necessary for the next cycle. This process is carried out for all contacts at each time cycle.

18. Suppose though, that a system of 100 disc elements, packed closely together, is to be considered. If the discs are of the same size, then there could, at most, be six contacts per disc; and since some must be on or near the edge, the total number of contact points must be less than 600. However, in order to check for contacts between the two arbitrary discs, it would (by brute force) be necessary to check for the possibility of a contact between each and every disc. Furthermore, an inordinate amount of searching would be required. If $n = 100$ discs, the number of different possible combinations C is given by

$$C = \frac{n!}{2(n-2)!} = \frac{100!}{2(98!)} = 4950 \qquad (14)$$

For 500 discs, the number of searches would increase to 124,750. In addition, computer memory space would be required at double the amounts above to store the normal and shear contact forces.

19. Since most of this memory would be blank and most of the searches fruitless (where contacts were not found), a great deal of effort was put into formulating schemes and methods to facilitate the program's efficiency. Indeed, the usability of the distinct element method is predicated on efficient programming techniques. Without incorporating these schemes it would not be possible to solve even small problems on large computers.

20. The various subroutines that make up the program DISC are related to each other primarily through named common storage. The items in common storage are as follows:

N = Number of disc and bar elements
SKN = Spring stiffness at contact
DKN = Damping constant of contacts
TIME = Time since beginning of run
DT = Time increment between steps
DF = Damping factor (a multiplier for the damping constant)
FXSUM(M) = Sum of forces in x-direction on element M
FYSUM(M) = Sum of forces in y-direction on element M
FMSUM(M) = Sum of moments acting on element M
FX(M) = Applied x-force on element M
FY(M) = Applied y-force on element M
FM(M) = Applied moment on element M
X(M) = Current x-coordinate of element M
Y(M) = Current y-coordinate of element M
T(M) = Current angular rotation of element M
R(M) = Radius (or thickness, if bar element) of element M
S(M) = One half of the separation between the circular ends of bar element M
W(M) = Weight of element M
XM(M) = Mass of element M
UX(M) = Current x-displacement of element M

UY(M) = Current y-displacement of element M

UT(M) = Current angular displacement of element M

VX(M) = Current x-velocity of element M

VY(M) = Current y-velocity of element M

IFIX(M) = Fixity code for element M . The values are:

1 = Fixed in x-direction

2 = Fixed in y-direction

3 = Fixed in x- and y-direction

4 = Fixed in angular direction

5 = Fixed in x and angular directions

6 = Fixed in y and angular directions

7 = Fixed in all directions

8 = A constant prescribed velocity applies

WGTF(M) = Weight factor (a multiplier for the initially assigned weights)

FRC(M) = Angle corresponding to coefficient of friction, i.e. tan(FRC) = μ

TEN(M) = Tensile strength at contact for element M

AA(M x 24) = AA is the contact list. See discussion of subroutine MOTION

NA(M) = Index used to find sought quantities in the contact list

LL (M x 6) = Possible contact search list. See discussion of subroutine PLIST

The common storage requires approximately 53 N storage locations.

Main Program

21. The main program initiates the graphics software and calls subroutine GIDYUP. Control is never returned to the main program. The graphics subroutines (West Point Military Academy, 1975) are peculiar to the WES computing system and also the Office of Personnel Management's Honeywell system at Macon, Georgia. The graphics subroutines employed in DISC include:

USTART	UBELL	UCRCLE
USET	UERASE	UPEN
UWAIT	UPRINT	UMOVE
UREAD	UPRNT1	UPSET
UGRIN	UARC	UDAREA
UWINDO	UROTATE	UOUTLN

Subroutine GIDYUP

22. Subroutine GIDYUP is the driving subroutine. Its primary function is to call subroutine INPUT and to cause cycling through subroutine OUTPUT. This subroutine is the only one that calls the main calculation subroutine MOTION. The normal sequence of operations involves:

a. GIDYUP calls INPUT. The user interactively describes the problem. Alternatively, the user may select a problem previously stored by subroutine SAVE.

b. A call is made to PLIST, which prepares a list of those elements for which a contact is possible within the next 50 cycles.

c. A call is made to MOTION, which calculates the movements of and forces on the elements. Fifty cycles are performed in MOTION. At the end of each 50 cycles, the new positions of the elements are graphically plotted by OUTPUT. Step b is again performed.

d. Every so often (user selected, default value = 250 cycles) a call is made to OUTPUT. The program operation is halted, and the user must inform the system what action is to be taken next (i.e., continue running, draw all elements, change parameters, etc.).

Subroutine INPUT

23. Subroutine INPUT is used to describe the problem to be analyzed (see user's guides (Figures 9 and 10)) and also to initialize common data storage dependent upon its calling subroutine. It may be called by GIDYUP (to start a new problem) or by OUTPUT (to modify an existing problem). The following subroutines may be called by INPUT:

SCREEN	CIRCLE
GIDYUP	LOOP
GRID	GEN
SAVE	

Subroutine OUTPUT

24. A variety of functions can be performed by subroutine OUTPUT (see user's guides (Figures 9 and 10)). After the problem geometry is described in INPUT, subroutine OUTPUT takes over the system operation. Its function is to set boundary conditions (fix certain elements), to allow for inputting parameters (weights, damping factors, frictional properties, etc.), and to handle both graphical and formatted output. Between the phases when calculations of motion are being performed between GIDYUP and MOTION, the program halts occasionally (user defined, default value = 250 cycles) in OUTPUT. During this halt, the user may redefine boundary conditions, parameters, etc., plot the data, or request another calculation phase.

25. During any program halt, the user may also instruct the program to store a copy of the existing data geometry (performed by subroutine SAVE). This stored data may then be recalled by a direct call to INPUT. Subroutine OUTPUT is also used to delete (erase) certain elements; however, elements may not be added in OUTPUT. At the outset of a problem, a great deal of geometrical editing may be accomplished by juggling the program back and forth between OUTPUT and INPUT. Consequently, the user is able to reposition elements, add elements, delete elements, change boundary conditions, etc.

26. The data storage (SAVE) feature is very handy for situations in which it is desired to solve a suite of problems with the same geometry but different parameters. After one problem is solved, the stored data may be recalled, the parameters changed, and the next problem solved.

27. Subroutine OUTPUT may be called only by subroutine GIDYUP. The following subroutines may be called by OUTPUT.

GIDYUP	CIRCLE	DAMP
INPUT	VECTOR	PLTFOR
LOOP	WEIGHT	GRID
SCREEN	INTERVAL	SAVE

28. Subroutine OUTPUT is also used to compute the contact stiffness k and time-step increment Δt . (Subroutine INPUT tentatively sets k and Δt in the same fashion as OUTPUT. However, OUTPUT redefines a stable time step dependent upon the boundary conditions.)

29. The equation of motion for an undamped linear oscillator of stiffness k and mass m is given by

$$m \frac{d^2u}{dt^2} + ku = 0$$

where u is the displacement. In a finice difference notation, the derivative d^2u/dt^2 may be expressed as

$$\frac{d^2u}{dt} = \frac{u_{t+1} - 2u_t + u_{t-1}}{\Delta t^2}$$

where u_{t+1} , u_t , and u_{t-1} represent the displacements at times $t+1$, t , and $t-1$, respectively. Substitution into the equation of motion results in

$$u_{t+1} + \left(\frac{k}{m} \Delta t^2 - 2\right) u_t + u_{t-1} = 0$$

The equality of the finite difference equation above can hold only if the quantity $\left(\frac{k}{m} \Delta t^2 - 2\right)$ is negative. Thus, the stable time step Δt is given by

$$\Delta t \leq \sqrt{\frac{2m}{k}} \tag{11}$$

Greater values for Δt will lead to exponentially increasing u .

30. It can be shown that the formulation for DISC (when considering a single normal contact) yields the same finite difference equation given above. That is, given (for some time step t) the values of u_t , Δu_t , v_t , and F_t in the normal directions, Equation 3 becomes

$$F_{t+1} = F_t - k\Delta u_t = -\sum_{t=0}^{t} k\Delta u_t = -ku_t$$

Equations 6 and 7 may be expressed as

$$v_{t+1} = v_t + \frac{F_{t+1}\Delta t}{m} = v_t - \frac{ku\Delta t}{m}$$

From Equation 8 and by substitution of the preceding equation

$$\Delta u_{t+1} = v_{t+1}\Delta t = v_t\Delta t - \frac{ku_t\Delta t^2}{m}$$

Then, making the substitution

$$v_t\Delta t = \Delta u_t = u_t - u_{t-1}$$

results in

$$\Delta u_{t+1} = u_t - u_{t-1} - \frac{ku_t\Delta t^2}{m}$$

Finally, Equation 9 may be written as

$$\begin{aligned} u_{t+1} &= u_t + \left(\Delta u_{t+1}\right) \\ &= u_t + \left(u_t - u_{t-1} - \frac{ku_t\Delta t^2}{m}\right) \\ &= -\left(\frac{k}{m}\Delta t^2 - 2\right) u_t - u_{t-1} \end{aligned}$$

Or by rearranging terms

$$u_{t+1} + \left(\frac{k}{m}\Delta t^2 - 2\right) u_t - u_{t-1} = 0$$

The displacement at time $t+1$ can be computed if the two previous displacements at times t and $t-1$ are known.

31. The procedure outlined for the computations of a stable time step Δt considered only one contact. In actuality, many contacts can be made for a single element; thus, the useful time step must be chosen smaller than the single contact time step. That is, most problems consist of many degrees of freedom.

32. Thus, a stable Δt will increase as the mass increases and will decrease as the stiffness increases. The program will, of course, calculate faster (i.e., the cost will go down) as the time step increases. Alternatively, if a Δt is chosen, then a stable k could be chosen for a given m. In general, any two of Δt, k, and m can be chosen, and the other computed to ensure stability through the preceding equation. The scheme incorporated into program DISC for computing a stable Δt is described in the following paragraphs. Any subsequent user of DISC should not necessarily feel bound to this scheme.

33. At the outset of writing DISC, it was decided that a disc element with a radius of 20 screen units would be the "standard" disc element. The Tektronix screen is 1023 screen units wide and 780 screen units high. For this standard element, a weight of 100.0 units is assigned. To determine the weight of other disc and bar elements, the area of the element is computed, divided by the standard area, and multiplied by 100.0. Thus, all forces output by the program are in terms of the 100.0 units assigned to the standard disc, i.e., the forces are normalized and, as such, are nondimensional. The element mass m_e is determined from

$$m_e = w_e/g \tag{12}$$

where

w_e = the normalized weight of the element

g = the acceleration due to gravity

The value g is an input parameter (default value = 32.2). Thus, if the default value of 32.2 is used, it implies an acceleration due to gravity of 32.2 screen units/unit of time/unit of time. If time is reckoned in seconds, this would normally suggest that one screen unit equals one foot (i.e., $g = 32.2\ ft/sec^2$). Inputting g as 9.8 (with time reckoned in seconds) suggests that one screen unit equals one metre (i.e., $g = 9.8\ m/sec^2$). It is through this input parameter g, that a scale can be assigned to the problem.

34. The contact spring stiffness k (currently set equal in normal and shear directions) can be assigned in a variety of ways. If the stiffness were physically known, it could be directly assigned. In most quasi-static problems, however, the purpose of the spring stiffness is to prevent the elements from penetrating too far into each other. This requirement implies choosing a very large k; however, choosing a large k results in a small Δt. For program DISC, the following equation is used for selecting k:

$$k = w_{max} \sqrt{N + 1} \tag{13}$$

where

N = the number of elements in the system

w_{max} = the weight of the largest element (actually the weight of the largest element not restrained from all movement or "fixed")

35. The time step Δt is then computed as

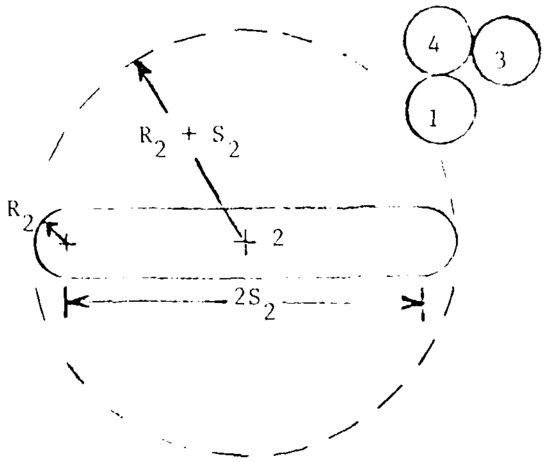
$$\Delta t = 0.05 \sqrt{m_{min}/k} \tag{14}$$

where m_{min} is the mass of the smallest element not restrained from all movement. The factor 0.05 is used to ensure stability since more than one contact per element is possible. In addition, subroutine OUTPUT computes element areas and moments of inertia and initializes many data lists.

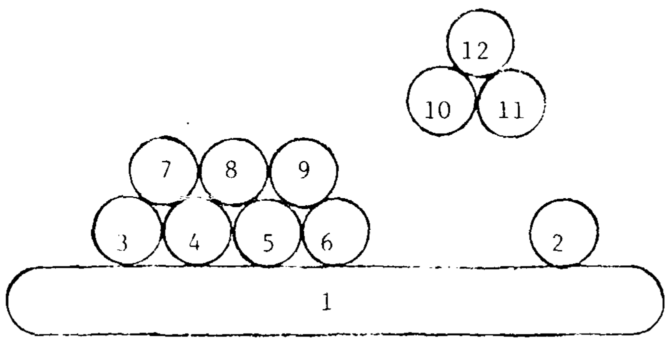
Subroutine PLIST

36. Subroutine PLIST is used to prepare a data list of possible contacts (between elements). This subroutine is accessed before each call to MOTION (i.e., every 50 cycles). Suppose the situation as shown in Figure 4 exists. To determine which elements would go into the possible contact list, a check is first made on the distance d_{ij} between each element, i.e.,

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{15}$$



a.



b.

Figure 4. Formation of possible contact list

where x_i , x_j , y_i , y_j are the coordinates of the element centroids. This distance is compared to the combined element thickness RNG_{ij} . The combined element thickness is the sum of the radii (plus lengths, if bar elements) of elements i and j , i.e.,

$$RNG_{ij} = R_i + R_j + S_i + S_j + 1 \text{ screen unit} \tag{16}$$

for a disc element $S_i = 0$.

One screen unit is added to RNG_{ij} so that elements that are quite close to contacting will be included in the possible contact list.

37. If $d_{ij} < RNG_{ij}$, then this contact pair is put in the possible contact list. If $d_{ij} > RNG_{ij} + 100$ screen units, the possibility of contact is rejected. If $RNG_{ij} < d_{ij} < (RNG_{ij} + 100)$, one further test is required to accept or reject the contact pair. Since a call to PLIST is made only every 50 cycles, it is possible that moving elements could make contact during the cycling interval. The velocity of each element (for the last computation cycle) is known at each call to PLIST. This velocity is used to compute an extrapolated position for the element at the end of the next 50 cycles. The original distance between elements i and j is given by d_{ij} . The extrapolated positions of elements i and j moving at velocities v_x^i , v_y^i and v_x^j , v_y^j are given by

$$\begin{aligned} x_i^e &= x_i + 50\Delta t v_x^i \\ y_i^e &= y_i + 50\Delta t v_y^i \\ x_j^e &= x_j + 50\Delta t v_x^j \\ y_j^e &= y_j + 50\Delta t v_y^j \end{aligned} \tag{17}$$

where x_j^e , y_i^e , etc., are the extrapolated element centroid coordinates after 50 time steps. By substituting

$$V_x = 50\Delta t(v_x^i - v_x^j)$$
$$V_j = 50\Delta t(v_y^i - v_y^j) \tag{18}$$

the new, extrapolated distance d_{ij}^e between element centroids becomes

$$d_{ij}^e = \sqrt{(x_i^e - x_j^e)^2 + (y_i^e - y_j^e)^2}$$
$$= \sqrt{(x_i - x_j + V_x)^2 + (y_i - y_j + V_j)^2} \tag{19}$$

Now, if $d_{ij} < d_{ij}^e$ (i.e., the original distance apart is less than the extrapolated distance apart), then the element pair is moving apart and the contact is not possible. However, if $d_{ij} > d_{ij}^e$, the element pair is closing and this pair is entered into the possible contact list provided that $d_{ij}^e \leq RNG_{ij}$.

38. At first sight, it would appear that the last test is all that is necessary to accept or reject the contact candidate. In fact, it is; however, to speed up computations, it is more efficient to check for possible contacts in the order described. As soon as a contact candidate is accepted or rejected, additional computations (involving the velocities) are no longer necessary. The idea is to accept or reject with as few computations as possible. It must be mentioned that PLIST forms only a possible contact list. This list is used in subroutine MOTION, where a closer examination is made to determine if contact is actually made. For example, as shown in Figure 4a, the bar element 2 obviously does not make contact with disc element 1. However, it will be included in the possible contact list. Also, elements 1 and 3 may, if they are moving together rapidly enough, be put in the possible contact list, but they may or may not physically come into contact during the next 50 cycles.

39. Searching for possible contacts is done in the following manner. Starting with element $i = 1$ and continuing incrementally to $i = n$ (the total number of elements), distance checks are made with all elements j with j ranging from $i + 1$ to n . Whenever a possible

contact with element i is found, the value j (the number of the element contacting i) is imbedded in the contact list. After all possible contacts with element i have been found, the value i is imbedded in the list to signal the end of all contacts with element i . For example, if it is assumed that elements 1 and 3, shown in Figure 4a, cannot make contact during the next 50 cycles, the possible contact list will be

2 - 4 - 1 - 2 - 4 - 3 - 4

Thus, element 1 makes possible contacts with 2 and 4, element 2 makes no new contacts (the 1-2 contact has already been found), element 3 makes contact with 4, and of course, element 4 could not possibly make any as yet undetected contacts. The underlined numbers, 1, 2, 3, and 4, are the ascending values of i and signify the end of each segment of the list. Now, if it is assumed that elements 1 and 3 could make contact, the list would be

2 - 3 - 4 - 1 - 2 - 4 - 3 - 4

For illustrative purposes, consider the arrangement of elements shown in Figure 4b. The possible contact list for the system would be

2 - 3 - 4 - 5 - 6 - 1 - 2 - 4 - 7 - 3 - 5 - 7 - 8 - 4 -
6 - 8 - 9 - 5 - 9 - 6 - 8 - 7 - 9 - 8 - 9 - 11 - 12 - 10 -
12 - 11 - 12

or a total of 19 contacts. Notice that the numbering is always forward. There is no reason to search backwards as those contacts will already have been found. The list segments are, in general, longer for the lower numbered elements. Within subroutine PLIST, the possible contact list is entitled LL (MLIST). The reserved length of LL is six times the number of elements, which should be adequate for most problems since this length will handle an average of six contacts per element. There is the possibility that the memory reserved could be exceeded if a large number of bar elements are present in the problem to be analyzed. Subroutine PLIST is called only by GIDYUP. No other subroutines are called by PLIST.

Subroutine MOTION

40. Subroutine MOTION performs three basic functions: (a) it detects element contacts, (b) it computes the forces acting on each element, and (c) it computes the subsequent displacement of the element due to those forces. Each entry to MOTION results in 50 time step iterations. Almost all of the processor time used during the operation of DISC is used in this subroutine.

41. Figure 5 is the flowchart for subroutine MOTION. As previously mentioned in the discussion of subroutine PLIST, a data list of possible contacts is created by PLIST before each call to MOTION. This list contains the elements that are presently touching or could be close enough to touch during the next 50 time cycles (i.e., the duration for each call to MOTION). While within MOTION, only those contact pairs contained in the possible contact list are considered as potential real contacts. It is not difficult to visualize that this list will be quite small compared to a list unrestricted by distance considerations. Indeed it is likely *(at least for disc elements)* that the possible contact list is a close approximation to the actual contacts.

Conditions for contacts

42. Figure 6 shows the various conditions for contacts between disc and bar elements. When describing element contacts, the convention is to state that element i contacts element j , where j is always greater than i . Searches for contacts are always forward, i.e., the lower numbered element is said to contact the higher numbered element, and never vice versa.

43. In general, four situations must be considered to detect real element contacts from the possible contact candidates i and j provided by PLIST.

a. Both elements i and j are disc elements (Figure 6a); only one contact can result.

b. Element i is a bar element and j is a disc element (Figure 6b); only one contact can result.

c. Element j is a bar element and i is a disc element (Figure 6c); only one contact can result.

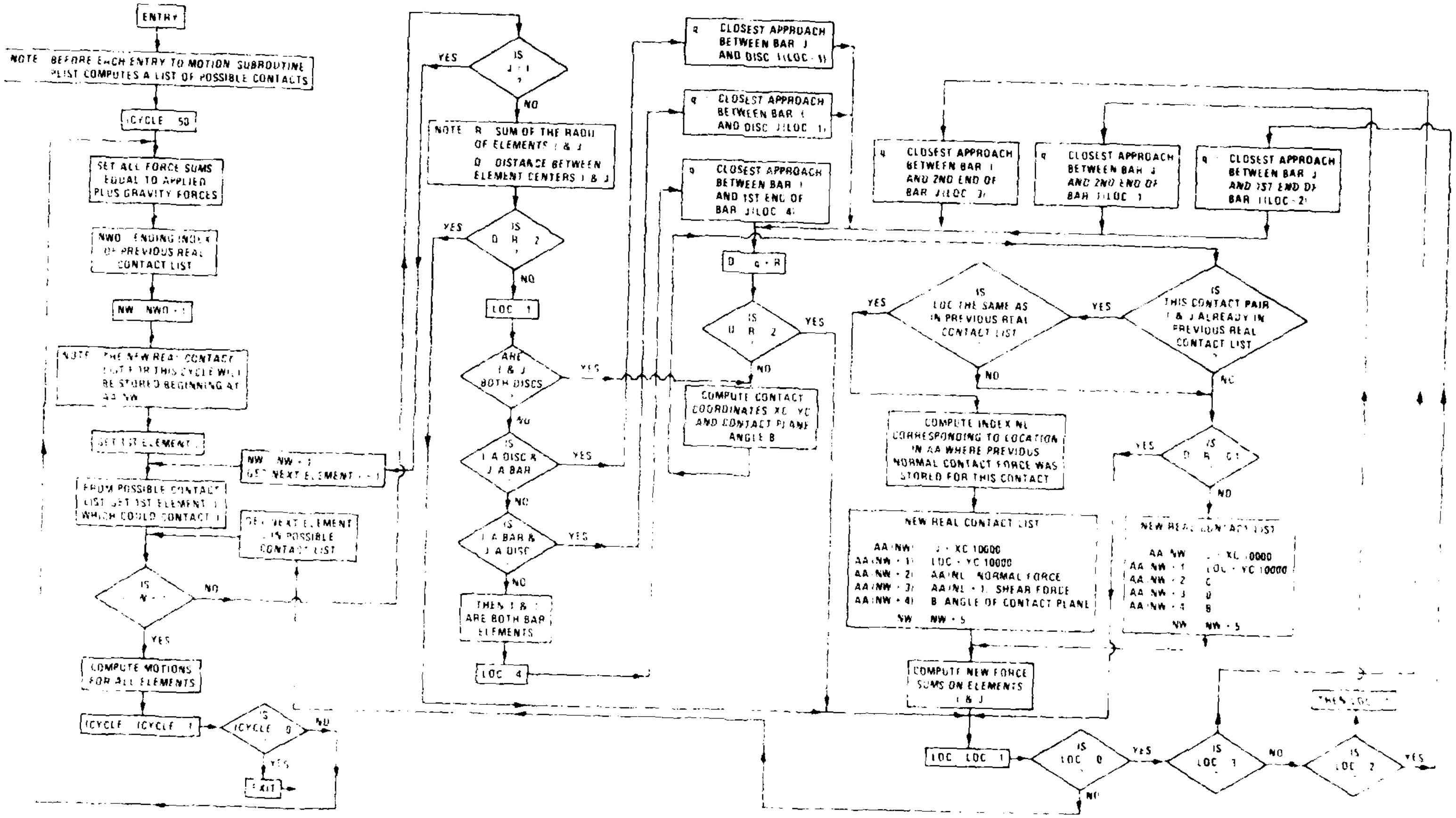


Figure 5. Flow chart of subroutine MOTION

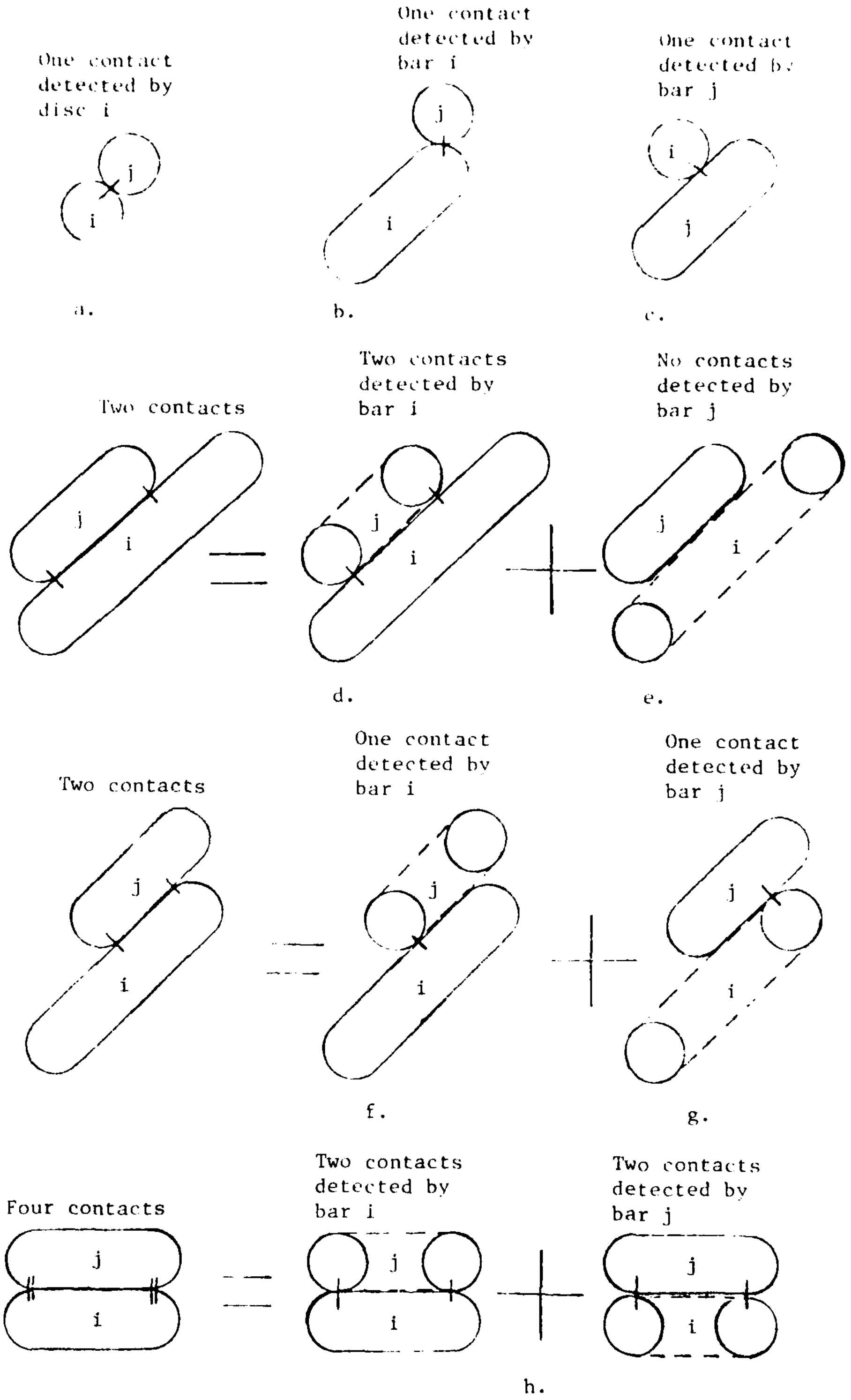


Figure 6. Detection of contacts

d. Both elements i and j are bar elements (Figures 6d, e, f, g, and h); one, two, or four contacts can result.

44. For situation a, *it is simply necessary to determine* if the disc elements are within the proximity criteria (to be discussed subsequently) in order to be real contacts.

45. For situation b, determine if disc element j is within the proximity criteria of the periphery of bar element i . If so, a real contact is detected.

46. For situation c, determine if disc element i is within the proximity criteria of the periphery of bar element j . If so, a real contact is detected.

47. For situation d, it is necessary to restrict the location of the points of contacts whenever two bar elements are parallel. For two parallel elements, the contact points are defined to be at the locations depicted in Figures 6d, e, f, g, and h, i.e., where circular ends of the elements become tangent to the central, straight portions. All of the contacts may be detected by considering two conditions: (1) element i to be a bar element and element j to be composed of two disc elements (*the discs being located at either end of bar* j), and (2) element j to be the bar element and element i to be composed of two disc elements.

48. Thus, as noted in Figure 6d, the necessary two contacts will be detected if i is considered to be the bar (condition (1) above) and j to be the two discs since both circular ends of bar j touch bar i . However, no contacts would be detected if j is the bar (Figure 6e, condition (2) above) since neither end of bar i makes contact with bar j . Figures 6f and 6g indicate that one contact is detected by choosing i as the bar and the other found by choosing j as the bar. Figure 6h represents the special case of two bars of equal length parallel to one another. In this case, four contacts (two coincident pairs) are detected; two from choosing i as the bar and two from choosing j as the bar.

49. For the situation in which two bar elements are not parallel, then only one contact between them is possible. However, the scheme just outlined for detecting contacts between bar elements is also used when examining skewed bar elements. The proximity criteria will reject any potential contacts not sufficiently close.

50. It must be mentioned that no matter whether element i or j or both are considered as the bar element (and the other as composed of two discs) when seeking contacts, the particulars about that contact are always associated with the lower numbered element, i.e., element i. This procedure is in keeping with the "going forward" rule.

Proximity criteria

51. The criteria for accepting or rejecting a possible contact are as follows:

a. If the peripheries of the two elements are within two screen units of each other at the subject contact point (recall that the "standard" disc element radius is 20 screen units), the contact is accepted as a candidate contact. Otherwise, it is rejected for this iteration cycle.

b. After a possible contact is elevated to candidate contact status, two more tests are necessary to finally accept the contact as a real contact. If this candidate contact is present in the contact list generated at the last time step cycle, it is accepted as a real contact and put in the new real contact list. If the candidate contact is not present in the previous contact list, then the two elements must be within 0.1 screen units of each other in order to be placed in the new real contact list. Otherwise, it is rejected for this iteration cycle.

Contact list

52. The contact list is a vector array in COMMON computer storage. This list contains the information about each contact that is necessary for further processing through subroutine MOTION. The contact array is defined by the named COMMON storage statement:

COMMON/STORE/AA(M x 24), NA(M)

Since M is the maximum number of elements (N is the actual number of elements) that may be considered for a problem, the size of AA is reserved to be at 24 times M. This size for AA should be adequate for most problems since only actual contacts are stored at any given step in the calculations. The contents of AA are constantly changing as the calculations proceed. The array AA should be thought of as divided into two segments: the first segment containing the contact information for

the just completed iteration (the previous contact list), and the second segment containing the information for the current contact list (or the new contact list). As contacts are detected, the contact information is embedded consecutively within the contact list. Since a given element can contact numerous other elements at any given time, the consecutive nature of the list is important to conserve memory. The array NA(i) given above is used to point to the location within AA where the contacts for a given element i begin.

53. Before describing the actual structure of the contact list, a discussion of the particular quantities going into the contact list is beneficial. For example, suppose a contact list is being built and the next available location within the array AA is AA(K), where K is the index within common storage. Further, suppose that a contact has just been detected between elements i and j and accepted as a real contact. Also, suppose that this same contact was present in the previous contact list starting at some location AA(L). Thus, the following information is put into the new contact list starting at AA(K):

AA(K) = $j + x_c/10000$

AA(K + 1) = ID + $y_c/10000$

AA(K + 2) = AA(L + 2), the previous normal contact force

AA(K + 3) = AA(L + 3), the previous shear contact force

AA(K + 4) = B, the angle from the horizontal to the contact plane

The symbols x_c and y_c are the coordinates of the point of contact, and ID is an identifier indicating the fashion in which the contact was detected (i.e., for all cases except two bar elements in contact, ID = 1 ; for two bar elements, ID takes on values of 1 to 4 depending in which sequence the contacts were found during the search). In effect, locations AA(K) and AA(K + 1) each contain two pieces of information. That is, AA(K) contains as its whole part the element number j (contacting element i) and as its fractional part x_c. A similar situation holds for AA(K + 1).

54. Strictly speaking, only the locations AA(K + 2) and AA(K + 3) contain information (the normal and shear forces on the contact) that

must be retained between iterations in order to accomplish the calculations for displacement. The need for storing the variables j and ID is to aid in determining if the contact now being considered is present in the previous contact list. The variables x_c , y_c , and B are retained in the contact list as an aid for later outputting of the results.

55. If it is now supposed that the contact being processed was not found present in the previous contact list, the new entry in the contact list would be the same as before, except

$$AA(K + 2) = 0.0$$
$$AA(K + 3) = 0.0$$

That is, the normal and shear contact forces would be set to zero.

56. Assuming that the contact between elements i and j just discussed was the first contact detected for element i , the contact list pointer NAA(i) is set equal to K . This pointer indicates the position in memory where the information about contacts for element i begins. Now that the contact data for this contact have been entered (and the entry required five memory locations), the next available location to store contact list data is determined by replacing K by K + 5 (i.e., K $\doteq$ K + 5). Any additional detected contacts for element i are, in this manner, put consecutively into the contact list; each detected contact requiring an additional five memory locations. The index K is incremented by five following each entry. After all contacts for element i have been detected and entered into the contact list (and also in the case of no contacts at all being detected for element i), the end of this contact list segment (for element i) is signified by embedding a zero into array AA at location K + 1 , i.e.,

$$AA(K + 1) = 0.0$$

The embedded zero indicates that this is the end of that segment of the contact list pertaining to element i . The next available location in array AA for storing contact data for the next element i $\doteq$ i + 1 is now located at K $\doteq$ K + 2 . Since the next element i is now ready to be processed, the pointer NA(i) is set equal to K . The just described process is repeated for all elements.

Initialization of contact data list

57. At the outset of each problem, it is assumed that there are no contacts between the various disc or bar elements. As mentioned in the preceding paragraph, zeros that are embedded in the contact list AA, at the location of the value of the element pointer NA(i) , indicate no contacts for element i . Therefore, at the beginning of a problem containing N disc and bar elements, the value of the pointer NA(i) is set to i yielding

$$\begin{aligned} NA(1) &= 1 \\ NA(2) &= 2 \\ &\vdots \\ NA(N) &= N \end{aligned}$$

Since a zero located at NA(i) within the array AA indicates no contacts, the array AA(K) is set to zero (for values of K = 1, N). In general, the first time cycle in MOTION will result in some contacts. The next available location to begin storing the first contact within AA is at N + 1 . The new pointer value for the first contact detected will be set to NA(i) = N + 1 .

58. Therefore, after each cycle of iteration the value of the pointer NA(i) is the location within the array AA where the contact list information about element i begins. If the element i has no contacts, the value AA(NA(i)) will be zero, signifying the end of the contact list segment for element i . If element i does possess contacts, the value AA(NA(i)) will be set to $j + x_c/10000$. To find the end of the contact list segment for element i , it is necessary to skip through array AA by increments of five (starting at AA(NA(i)) until a zero is found. The next available location in AA will possess information about element i + 1 .

Example of
contact list information

59. Consider the simple example shown in Figure 7a. At the start of the problem, two disc elements (2 and 3) are at rest on bar

element 1. An additional disc element (5) is resting upon bar element 4. Both bar elements are assumed to be fixed, i.e., they are prevented from rotation and translation. Assuming that the only forces acting on this system are due to gravity, the subsequent motions of the disc elements may be visualized according to the sequence indicated by Figure 7. Figure 8 shows the structure of the contact list for various stages of movement.

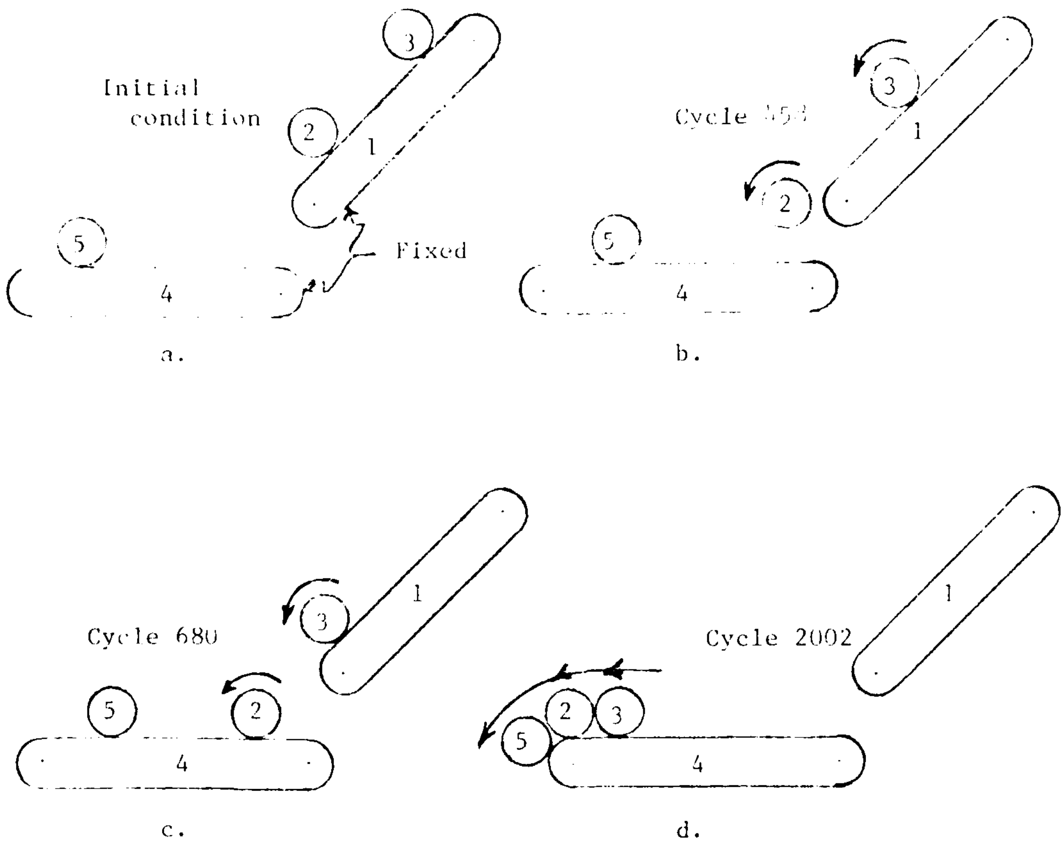


Figure 7. The contact list

Figure 8. Organization of the contact lists

60. As previously mentioned, each call to subroutine MOTION is preceded by a call to subroutine PLIST. Subroutine PLIST creates the possible contact list LL. The list LL as given at the top of Figure 8 is for the conditions shown in Figure 7a. The interpretation of the list is as follows: elements 2 and 3 may contact element 1, element 2 has no (as yet undetected) contacts, element 3 has no (as yet undetected) contacts, element 5 may contact element 4, and element 5 has no undetected contacts.

61. At the outset of any problem, the pointer list NA(I) is initialized to the value of I and the contact list is set to zero. Thus, initially, NA (element 1) = 1 , NA(2) = 2 ,.....NA(5) = 5 , and the contact data list (list 1, Figure 8) contains zeros as indicated by the symbol E (for empty). Since there are five elements in the problem, the first five storage locations are set to zero (as indicated by the five "E's" in list 1). The next available location for contact storage (at the next, or in this case the first cycle) is location 6.

62. The situation for the first cycle of MOTION is shown in list 2 (Figure 8). The pointer values NA are: NA(1) = 6 , NA(2) = 17 , NA(3) = 18 , NA(4) = 19 , and NA(5) = 25 . Location 6 of list 2 is the beginning of the contact information for contact between elements 1 and 2. Immediately following the contact information for contact 1-2 is the information for contact 1-3 (starting at location 11 and ending at location 15). The zero (symbol E) embedded at location 16 signifies that there are no more contacts for element 1. Contact information for element 2 begins in the contact list at location 17 (NA(2) = 17). Location 17 contains a zero, which indicates that there are no contacts for element 2. Similarly, no contacts exist for element 3. The pointer for element 4 (NA(4) = 19) points to a nonzero value. Thus, the next five memory locations (19-24) yield contact information for contact 4-5. The pointer for element 5 (NA(I) = 25) points to a zero; therefore, there are no new contacts for element 5. Indeed, the last element may never possess any contact not already found.

63. A total of 20 memory locations were required to store all of the contacts for cycle 1. Since storage began at location 6, at present

25 locations have been used. Thus, the next available location for storage is 26.

64. At any given time cycle, it is necessary to retain the contact information about the previous time cycle. Only by retaining this information is it possible to check whether or not a detected contact was present at the last stage or if the contact is new. (If the contact is old, the normal and shear forces for the previous cycle are entered into the new contact list; otherwise, zero values for shear and normal forces are entered.) Figure 8 gives the new contact list for cycle 2 (list 3). List 3 is exactly the same as list 2 except the new contact list begins at location 26 and ends at location 45. It is readily apparent that all of these "new" contacts entered into the new contact list were present at the previous cycle. The pointer NA for a particular element is not updated to the new location until after it is used to check whether or not this same contact was present in the previous list. For example, contact 1-2 has been identified as a contact for cycle 2, and the contact information is to be entered at locations 26 through 30. The pointer value for element 1, NA(1) , still has a value of 6. By going to location 6, it is noted that contact 1-2 was present in the previous contact list. The pointer value NA(1) is temporarily updated to a value of 11 so as to be directly pointing to the next location in the previous contact list where contact information about element 1 was stored. Thus, when contact 1-3 is examined (as to whether or not it was in the previous list), the pointer is in the proper position for quick recognition of this preexisting contact. After it has been determined that all contacts for element 1 have been exhausted, the pointer NA(1) is set to the new, current value of 26.

65. At each and every cycle an examination is completely carried out for every contact given in the <u>possible</u> contact list. Information stored for the previous cycle is not used at all for determining whether

or not a contact is entered into the new contact list.* Only after it has been decided to enter a contact into the list is a scanning of the previous list required (and then only to decide what to do with the shear and normal forces). The method for updating the element pointer NA(I) simply provides an efficient method for checking on the presence of the contact in question.

66. The situation for cycle 3 is shown in list 4. Again, the organization is the same as for cycles 1 and 2 except the new contact list is removed by 20 more locations (i.e., locations 46 through 65). The situation for cycle 4 is not shown, but would be similar, except the new contact list would be stored in locations 66 through 85. This process could be carried out indefinitely; however, at some point memory would become exhausted. Since it is necessary to store contact information only for the previous and present cycles, a scheme was developed for recovering all available memory.

---

* There is an option available that precludes the searching for contacts at each and every cycle. By entering a suitable variable (Command I in the Input Phase (Figure 9)), it is possible to stipulate that complete searches for contacts will be made only after every so many cycles. Unless otherwise stipulated through these options, a complete search is made for those element contacts in the possible contact list at each cycle. If the option is used, the program assumes that the current real contact list is to be used over and over; the shear and normal forces are being constantly updated. During this time, the contact coordinates and contact plane angles are properly computed. The use of this option can be dangerous since new contacts will not be detected until a complete search is called for. A complete search is always made upon each entry into MOTION. The use of this option can significantly reduce computer processing time since it negates the need for contact searches and for creating a new contact list.

67. Suppose that 120 storage locations* have been reserved for contact information as indicated in Figure 8. While processing in cycle 1, it is apparent that locations 1 through 25 are needed for the present and the previous contact lists. Thus, locations 26 through 120 are available as "free" memory for future cycles. While in cycle 2, the required memory locations are 6 through 45 (old plus new contact list). Therefore, the free memory consists of locations 46 through 120 plus 1 through 5. At cycle 3, the free memory consists of locations 66 through 120 plus 1 through 25. During cycle 4, the free memory extends from locations 86 through 120 plus 1 through 45. At cycle 5, the free memory consists of locations 106 through 120 plus 1 through 65. Thus, as cycle 6 is entered (and will require 20 locations to store the new contact list), it is apparent that the reserved memory space of 120 will

---

* The size of common memory for the variables used in program DISC is set through the use of a Fortran PARAMETER statement. As seen from an examination of the program listing, the space allocated for the contact list AA is set to $M \times 24$, where M is the maximum number of elements permitted for a problem. In practice, M will often be greater than the actual number of elements being considered, thus extra memory space is generally allocated. The common dimensions of $M \times 24$ should be adequate for most problems. For example, consider a collection of closely packed uniform disc elements, x discs wide by y discs high. The total number of elements is xy. The maximum total number of contacts for this collection may be shown to be $3xy - 2(x + y) + 1$. Since five storage spaces are required for each current contact (plus one additional space for each contact to store the zero or "E" marks) the storage S required to contain the current contact list is

$$S = 5[(3xy - 2(x + y) + 1)] + xy$$
$$= 16xy - 10(x + y) + 5$$

Neglecting the last two terms, S equals 16 times the total number of elements. Technically, twice this amount of storage is required at any given cycle since the last contact list must be scanned to prepare the current contact list. In actuality, twice the amount is not needed because as soon as the new contacts are entered into the current contact list, there is no longer any need for the corresponding previous contact (as long as the sequence of insertion remains the same). At present, it is felt that a reserved space of $M \times 24$ should be adequate for most problems.

be exceeded unless the front part of memory is reused. This condition can be avoided by computing the amount of free memory still available up to location 120 after each entry into the contact list. Since the next entry could require five additional storage locations, the beginning of the next entry is directed to location 1 if sufficient memory does not exist. Therefore, as shown for cycle 6, the first detected contact 1-2 is placed in locations 106 through 110, and the next contact placed in locations 111 through 115. The five storage locations available before location 120 are sufficient for storage of one more contact. However, contact 1-3 was the last contact for element 1. Therefore, a zero ("E") is placed at location 116, signifying the end of contacts for element 1. There are now only four available locations before the end of the reserved memory and, therefore, not enough locations to store information for a detected contact. The program now directs that storage begin at location 1. As shown for cycle 6, a zero ("E") is placed in location 1 (signifying that element 2 has no, as yet, undetected contacts). The pointer NA(2) is set to the value of 1. Locations 117 through 120 are not used.

68. As long as the contacts do not change, the calculations will proceed as described in cycles 1 through 6, each cycle requiring an additional 20 new storage locations of free memory. Cycle 457 is the situation just before element 2 rolls off bar element 1. Figure 8 presents the possible contact list existing for the conditions shown in Figure 7b. The interpretation of this list is as follows: elements 2 and 3 may contact element 1, element 4 may contact element 2, element 3 has no undetected contacts, element 5 may contact element 4, and element 5 has no new contacts. The possible contacts 1-2 and 2-4 are included in the possible contact list; however, they will be excluded as real contacts in MOTION since the situation shown in Figure 7b is for the condition where element 2 is just breaking contact with element 1 and before contact with element 4. Figure 8 also shows the new contact list for cycle 458. Note that there are now only two contacts (contacts 1-3 and 4-5). The contact list for cycle 459 is just as it was for cycle 458 except the new contact is stored an additional 15 locations further

along in memory. Cycle 679 is the situation just before element 2 hits element 4. In addition, Figure 8 presents the possible contact list and the resulting new contact lists (cycles 680, 681, and 1085). Notice that again there are three contacts (contacts 1-3, 2-4, and 4-5).

69. In Figure 8, the possible contact list and the contact data list are given for some much later time (say cycle 2002). These lists are for the conditions depicted in Figure 7d. By this time, all of the disc elements are in contact and proceeding to roll off the end of element 4. At this stage, five contacts are detected (contacts 2-3, 2-4, 2-5, 3-4, and 4-5). Thus, 30 additional storage locations are required to store the contacts at each cycle.

Calculations of forces on the elements

70. As each element is, in turn, examined for contacts, the sum of all forces acting on the element is computed. Preceding each time cycle of iteration the force sum is set equal to the applied plus gravity loads, i.e.,

F^i_{ysum} = vertical applied load minus the gravity load

F^i_{xsum} = horizontal applied load

M^i_{sum} = applied moment

In the event that the element being examined possesses no contacts, the computation proceeds directly to the calculation of accelerations and displacements (see Equations 6 through 9, Part II). If contacts are found for the element being considered, then the force sums are modified in accordance with Equations 1 through 5. The force sums are updated for both elements in contact. For example, suppose a situation exists in which element 3 makes contact with elements 5 and 7. When element 3 is examined for its contacts, the force sums will be updated twice (once for element 5 and again for element 7). During this process, the force sums for elements 5 and 7 are also updated. Then when element 5 is examined and it is determined that it has no, as yet, undetected contacts, the computation will proceed directly to Equation 6 since the force sum for element 5 was updated at the time when element 3 was examined. The same holds for element 7.

71. The computations of Equations 8 and 9 are particularly important. Equation 8 yields the new incremental element centroid displacements, and Equation 9 yields the new coordinates of the element centroid. It is these quantities that are required by Equation 1 for the next iteration cycle.

Other Subroutines

Subroutine PLTFOR

72. Subroutine PLTFOR is used to output the forces acting at the contacts between elements. Recalling that the contact list is structured such that each element has at least one entry (if no contacts were detected) and further that the storage is in sequential order, it is possible to, at any stage, thread through the contact list to obtain the contact forces. For each element i that possesses contacts, the contact list contains:

a. The number j of the element being contacted by i.

b. The ID number for the contact (two bar elements in contact may have up to four contact points).

c. The contact point coordinates, x_c and y_c.

d. The normal and shear forces acting at the contact, F_n^c and F_s^c (stored as AA(K + 2) and AA(K + 3) in memory).

e. The angle to the contact plane, B.

73. These forces are output graphically. A pass is first made through the contact list to determine the largest absolute value of the shear or normal force in the list. This quantity is used to calculate a scale factor, and the vector representing this force will be 20 screen units long. The other vectors drawn to represent the shear and normal forces will be proportional. Thus, after each call to PLTFOR, a graphical representation of the contact forces are drawn on the screen. The contact coordinates x_c and y_c are used to locate the position of the vectors to be drawn, the shear force is drawn at the angle B, and the normal force perpendicular to the shear force. The user also has the option of obtaining a printout of the contact forces (Figure 9 or 10).

Subroutine LOOP

74. Subroutine LOOP is used to check whether or not the cross-hair cursor is located in the vicinity of an element centroid. In some cases, this test is used to modify certain commands.

Subroutine INTERVAL

75. Subroutine INTERVAL is used to change the frequency of returning control to the terminal. The normal method is to perform 250 time steps of calculations before halting calculations. At each halt the user directs what action is to be taken next.

Subroutine GRID

76. Subroutine GRID causes a grid to be drawn on the screen.

Subroutine SAVE

77. Subroutine SAVE may be called at any time by OUTPUT to store the current element geometry in memory. Subroutine SAVE is automatically called on each exit from INPUT. This feature provides the user a means of reworking the same geometry with, perhaps, a different set of parameters.

Subroutine CIRCLE

78. Subroutine CIRCLE aids in the graphical output of disc and bar elements.

Subroutine SCREEN

79. Subroutine SCREEN is used to position the alpha cursor on the screen following input queries and some output functions. It prevents overwriting certain areas of the screen.

Subroutine GEN

80. This subroutine may be called during the INPUT phase. Its function is to create a collection of disc elements touching one another in a dense packing.

Subroutine VECTOR

81. This subroutine is used to vary the form of the graphical output. The normal mode of display is to draw the elements on the screen at their current positions. Subroutine VECTOR causes the display to track the element centroids.

Subroutine WEIGHT

82. Subroutine WEIGHT may be used to modify the weight of any or all elements.

Subroutine DAMP

83. Subroutine DAMP is used to change the damping constant K .

PART IV: PROGRAM OPERATION

84. The computer program DISC was written in such a fashion as to keep the data preparation simple and minimal. As a result, use of computer graphics was employed to a large extent. Most of the commands necessary to operate DISC consist of single keystrokes at the Tektronix terminal. Figures 9 and 10 are the computer user's guides for the operation of DISC. Figure 9 gives a more detailed discussion of the user commands; Figure 10 is an abbreviated command listing. When one first uses the program DISC, the information given in Figure 9 should guide the user in the operation of the program. After the user becomes familiar with the program, the abbreviated command list (Figure 10) should suffice.

85. According to the user's guides, all input is initiated by entering a single character, following the appearance of the crosshair cursor. The crosshair cursor consists of two perpendicular lines crossing the screen of the terminal. The position of the intersections of the two lines is controlled by a set of thumbwheels on the terminal. In many instances, the entry of the single character is sufficient to accomplish whatever is desired. In other instances, certain parameters (or numbers) associated with the single character command are required. If the command requires one or more parameters, the alpha cursor (a small matrix of dots) will appear on the screen. As soon as the alpha cursor appears, the input parameters may be typed.

86. The detailed user's guide (Figure 9) is complete regarding the meaning of the various commands and, as such, will not be discussed further. Three examples will, instead, be described in order to illustrate the program.

87. Figure 11 is a series of computer drawings that describe a very simple problem. As soon as program DISC is called into execution, the message "INPUT PHASE" is written on the screen (Figure 11a). The collection of three elements was created in the following order:

```
*********** INSTRUCTIONS FOR OPERATION OF PROGRAM 'DISC'   **************
*                                                                       *
*                    Commands for Input Phase                           *
*                   --------------------------                          *
*                                                                       *
*  All commands are entered after the appearance of the crosshair cursor*
*  Parameters, if required, are entered after appearance of the alpha   *
*  cursor which will appear on the right side of the screen.            *
*                                                                       *
* KEY---Parameters                  Description                         *
*------------------               -------------                         *
*                                                                       *
* [space]---none              Creates disc element centered at crosshair*
*                             cursor position                           *
*                                                                       *
* 1                           1st end of a bar element ( semi-circle    *
*                             centered at crosshair)                    *
*                                                                       *
* 2                           2nd end of a bar element. Note: repeated  *
*                             hitting of the 2 key will create bar elements*
*                             touching the the 2nd end of the preceeding*
*                             bar element.                              *
*                                                                       *
* A                           Creates a disc element adjacent to a      *
*                             preceeding disc element in direction      *
*                             indicated by the crosshairs               *
*                                                                       *
* B---(3)                     Generates a Batch of disc elements touching*
*                             each other configured by entering         *
*                             parameters. The 1st element is centered at*
*                             the crosshairs.                           *
*                             Parameters-----                           *
*                              1st-- # of discs in bottom row           *
*                              2nd-- # of rows                          *
*                              3rd-- increment for more or less discs(+or-)*
*                                    for each successive row upward     *
```

Figure 9. Detailed instructions (Sheet 1 of 6)

```
*                                                                          *
*   D                         Draws all elements. Screen will first erase  *
*                             and then redraw all elements.                *
*                                                                          *
*   C                         Prints Coordinates of indicated element      *
*                                                                          *
*   R                         Changes value for Radius of discs and        *
*                             width of bar elements. Default value=20.     *
*                             screen units.                                *
*                                                                          *
*   J---(2)                   Moves the J-th element to new position       *
*                             given by paramerers (x-y coordinates).       *
*                             Redisplay by hitting D key                   *
*                                                                          *
*   P---(2)                   Re-Position all elements to new position     *
*                             (given by parameters) relative to element    *
*                             indicated. Screen automatically erases       *
*                                                                          *
*   + or;                     Draws a Grid on the screen.                  *
*                                                                          *
*   E                         Ends input phase. Calls compute phase. Also  *
*                             causes a copy of the current geometry to be  *
*                             stored in memory                             *
*                                                                          *
*   I---(1)                   Sets interval between complete searches to   *
*                             detect contacts. Default value=1, maximum    *
*                             value =50                                    *
*                                                                          *
*   [backspace]               Recalls & displays the last stored copy      *
*                                                                          *
****************************************************************************
*                                                                          *
*                      Commands for Compute Phase                          *
*                      --------------------------                          *
*                                                                          *
*     The following commands are effective regardless of the position     *
*          of the crosshair cursor.                                        *
*                                                                          *
```

Figure 9. (Sheet 2 of 6)

```
*   Key--parameters                    Description                              *
*   ---------------                    -----------                              *
*                                                                               *
*  G                            GO- Calculate motions. Program calculates       *
*                               the # of cycles given under I command (below*
*                               Default interval is 250 cycles                  *
*                                                                               *
*  I---(1)                      Sets Interval between program interrupts.       *
*                               Program calculates the given # of cycles        *
*                               before inquiring as to what action is to be     *
*                               taken next. Default value is 250 cycles         *
*                                                                               *
*  N                            New problem. Screen erases, all memory is       *
*                               set to zero. Returns to input phase.            *
*                                                                               *
*  S                            Saves current geometry and fixities.            *
*                               (Replaces the stored copy which was             *
*                                automatically saved upon exit from input       *
*                                phase.                                         *
*                                                                               *
*  Q---(1)                      Damping factor-Parameter is % of critical       *
*                               damping. Default value is 1.0                   *
*                                                                               *
*  Z                            Zeroes all element velocities.                  *
*                                                                               *
*  C                            Draws a clock on the screen giving time from*
*                               start of problem                                *
*                                                                               *
*  f                            Fixes all elements from movement                *
*                                                                               *
*  K                            Erases (kills) screen. Reset page can be        *
*                               used for same, however K causes the command     *
*                               to be computer generated--handy if copying      *
*                               to a disc or tape device.                       *
*                                                                               *
*  D                            Draws all elements                              *
*                                                                               *
*  O(letter)--[where]           Causes vector plot of contact forces to be      *
```

Figure 9. (Sheet 3 of 6)

```
*                                    displayed. If printed output is desired,      *
*                                    position tne crosshair to the upper right     *
*                                    side of [where] printing is to start.         *
*                                    Then hit the H key. Hitting any other key     *
*                                    will give plot only                           *
*                                                                                  *
*   * or :                           Causes output to be in vector mode, i.e.      *
*                                    instead of circles & bars being drawn, lines*
*                                    connectig successive element centroid        *
*                                    locations are drawn. Repetition of * key      *
*                                    alternates modes back & forth.                *
*                                                                                  *
*   + or ;                           Draws a Grid on the screen.                   *
*                                                                                  *
*   E                                Ends compute phase. Allows user to return     *
*                                    to input phase for purposes of recalling      *
*                                    a stored problem or to add elements. User     *
*                                    returns to compute phase by again hitting     *
*                                    E key (in input phase).                       *
*----------------------------------------------------------------------------------*
*                                                                                  *
*     The following commands are effective ONLY when the crosshair is              *
*     centered (located within 5 screen units) on the centroid of the              *
*     element for which the effect is desired.                                     *
*                                                                                  *
*    Key--parameters                       Description                             *
*    ---------------                       -----------                             *
*                                                                                  *
*   F                                Fix element (from moving) in all directions *
*                                     X-Y-& angular.                               *
*                                                                                  *
*   X                                Fix element from movement in X-direction.     *
*                                                                                  *
*   Y                                Fix element in Y-direction                    *
*                                                                                  *
*   M                                Fix element in angular (moment) direction     *
*                                                                                  *
*   U                                Unfix the element. Note: if any element has *
```

Figure 9. (Sheet 4 of 6)

```
*                                   previously been fixed in one or more         *
*                                   modes, and it is desired to change that      *
*                                   element to a different mode, first Unfix     *
*                                   the element before assigning the new fixity  *
*                                                                                *
*   [space]                         Erase (destroy) the indicated element from   *
*                                   the system. Elements are re-numbered.        *
*                                                                                *
*   L---(3)                         Apply a load to a element at it`s centroid   *
*                                    1st parameter-X load                        *
*                                    2nd parameter-Y load                        *
*                                    3rd parameter-Moment                        *
*                                     Note: Each unit of load or moment is       *
*                                           equivalent to the weight or moment   *
*                                           of inertia of the standard 20        *
*                                           screen unit radius circle            *
*                                                                                *
*   V---(3)                         Sets INITIAL velocity in each direction.     *
*                                   In screen units per second                   *
*                                    1st parameter-X velocity                    *
*                                    2nd parameter-Y velocity                    *
*                                    3rd parameter-Rotational velocity           *
*                                                                                *
*   v---(3)                         Sets velocity to prescribed CONSTANT value   *
*                                     Parameters same as above.                  *
*                                                                                *
*   c                               Prints coordinates of indicated element.     *
*                                     Parameters same as above.                  *
*                                                                                *
*   ? or /---[where]                Causes centroid information to be printed    *
*                                   on screen. After hitting ? key, position     *
*                                   crosshair [where] output is to be printed.   *
*                                    Output is of the form:                      *
*                                    El no.----X force-----Y force---Moment      *
*                                    X vel-----Y vel-------Ang vel               *
*                                    Fix code--Weight fac--Fric val--tension val*
*--------------------------------------------------------------------------------*
*                                                                                *
```

Figure 9. (Sheet 5 of 6)

```
*     The following commands may be interpreted in two different ways.    *
*      If the crosshair cursor is positioned on an element centroid,then  *
*      ONLY that element is affected by the command. If the crosshair is  *
*      NOT positioned on ANY centroid, the command applies to ALL elements*
*                                                                         *
*    Key--parameters               Description                            *
*    ---------------               -----------                            *
*                                                                         *
*   0(zero)---(1)           Angle of friction in degrees for element.     *
*                           (u=tan φ). If two contacting elements have
*                           different friction angles, the coefficient    *
*                           associated with the smallest angle is used.   *
*                           Negative angles can be entered. If so, the    *
*                           coefficient is computed as the absolute       *
*                           value of the most negative angle.             *
*                           For example---if el # 1 has φ=0' and el # 2
*                           has φ=-30', then the coefficient of friction
*                           between el #1 &el #2 is tan(30').             *
*                           Default value is φ=26.56', u=0.5
*                                                                         *
*   T---(1)                 Tension value (force units) between elements*
*                           Default value is 0.0. Same rule for negative*
*                           values applies as for 0.                      *
*                                                                         *
*   W---(1)                 Weight factor for element(s). Used to         *
*                           increase or decrease weight or to make all    *
*                           elements weightless. The mass is not changed*
*                           Default value is 1.0.                         *
*                                                                         *
*   ^-----^                 Used to determine distance & angle between    *
*                           two points. Hit ^ key at 1st point, then      *
*                           hit ^ (or any key) at 2nd point. Distance     *
*                           and angle (degrees) between them is printed.*
*                           If crosshair is on/near an element centroid   *
*                           for either hit, the element centroid coor-    *
*                           dinates are used exactly.                     *
*                                                                         *
***************************************************************************
```

Figure 9. (Sheet 6 of 6)

ABBREVIATED COMMAND INSTRUCTIONS FOR "DISC"

Input phase

Key-parameters	Description
[space]	Creates disc element
1	1st end of bar element
2	2nd end of bar element
A	Places disc element Adjacent to preceeding
B---(3)	Creates Bunch of disc elements
D	Draws all elements
C	Prints coordinates of element
R	Change Radius of disc (width of bar).
J---(2)	Moves J-th element
P---(2)	Re-Positions all elements
+ or ;	Draws a Grid
E	Ends input phase. Creates a stored copy.

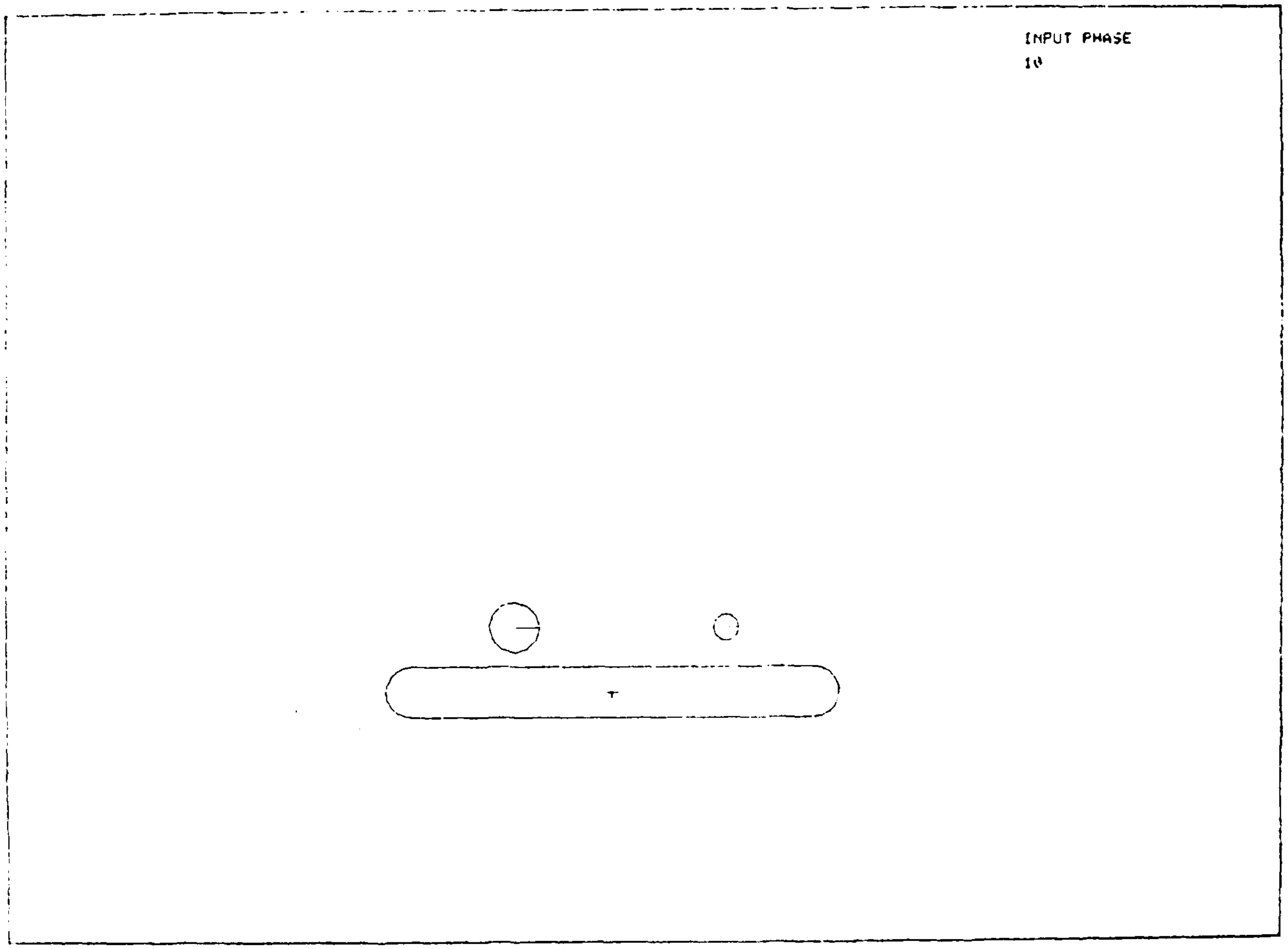
Figure 10. Abbreviated instructions (Sheet 1 of 3)

```
*   I---(1)                         Sets interval between contact searches.    *
*                                        -                                     *
*   [backspace]                     Recalls last stored copy.                  *
*                                                                              *
*------------------------------------------------------------------------------*
*------------------------------------------------------------------------------*
*                        Compute phase                                         *
*                        -------------                                         *
*                                                                              *
*   Key-parameters                  Description                                *
*   --------------                  -----------                                *
*                                                                              *
*   G                               Go--calculate motions                      *
*                                   -                                          *
*   I---(1)                         Set Interval between queries ( #-cycles)   *
*                                       -                                      *
*   N                               New problem. Restart from scratch.         *
*                                   -                                          *
*   S                               Save (store) a copy of present geometry    *
*                                   -                                          *
*   Q---(1)                         Damping factor.                            *
*                                                                              *
*   Z                               Set all velocities to Zero.                *
*                                                         -                    *
*   C                               Draws clock on screen.                     *
*                                         -                                    *
*   f                               Fixes all elements.                        *
*                                                                              *
*   K                               Erase (Kill) screen.                       *
*                                          -                                   *
*   D                               Draws all elements.                        *
*                                   -                                          *
*   O (letter)--[where]             Output contact forces.                     *
*                                   -                                          *
*   * or :                          Output to vector mode and vice versa       *
*                                                                              *
*   + or ;                          Grid is drawn.                             *
*
```

Figure 10. (Sheet 2 of 3)

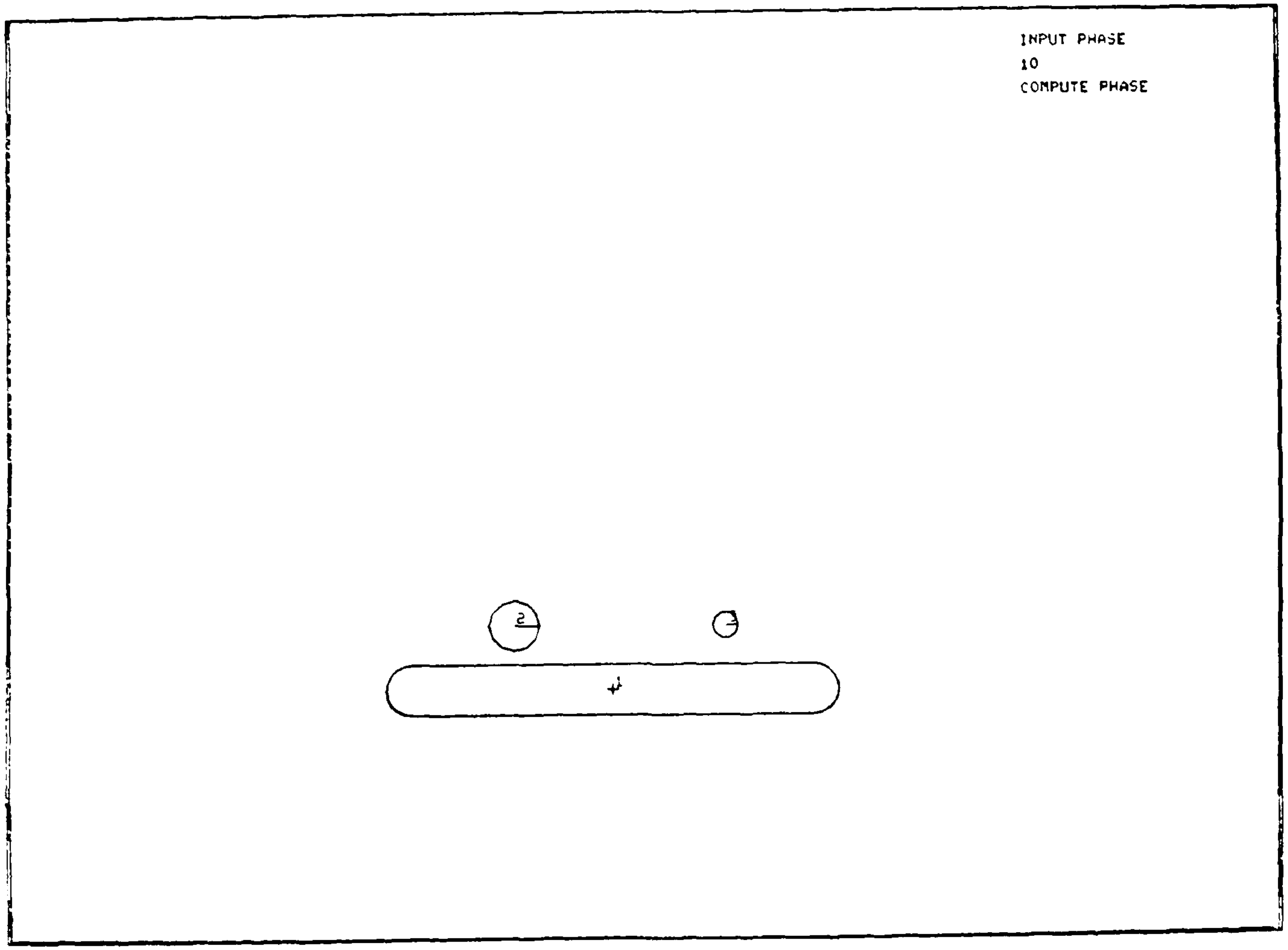
```
*    E                        Ends compute phase. Calls input phase.     *
*                             -                                          *
*    F                        Fix element in all directions.             *
*                             -                                          *
*    X                        Fix in X-direction.                        *
*                                    -                                   *
*    Y                        Fix in Y-direction.                        *
*                                    -                                   *
*    M                        Fix angular rotation.                      *
*                                                                        *
*    U                        Unfix element.                             *
*                             -                                          *
*    [space]                  Erase (destroy) element                    *
*                                                                        *
*    L---(3)                  Enter Loads--X-Y-Moment.                   *
*                                   -                                    *
*    V---(3)                  Enter initial velocities--X-Y-Angular      *
*                                   -------- -                           *
*    v---(3)                  Set constant prescribed velocities         *
*                                 --------           -                   *
*    c                        Print element centroid coordinates.        *
*                                   -------                              *
*    ? or / ---[where]        Print centroid information.                *
*                                                                        *
*    0 (zero)---(1)           Enter Friction angle- 0                    *
*                                                   -                    *
*    T---(1)                  Enter Tension value.                       *
*                                   -                                    *
*    W---(1)                  Enter Weight factor.                       *
*                                   -                                    *
*    ^-----^                  Print Distance between points.             *
*                                                                        *
**************************************************************************
```

Figure 10. (Sheet 3 of 3)



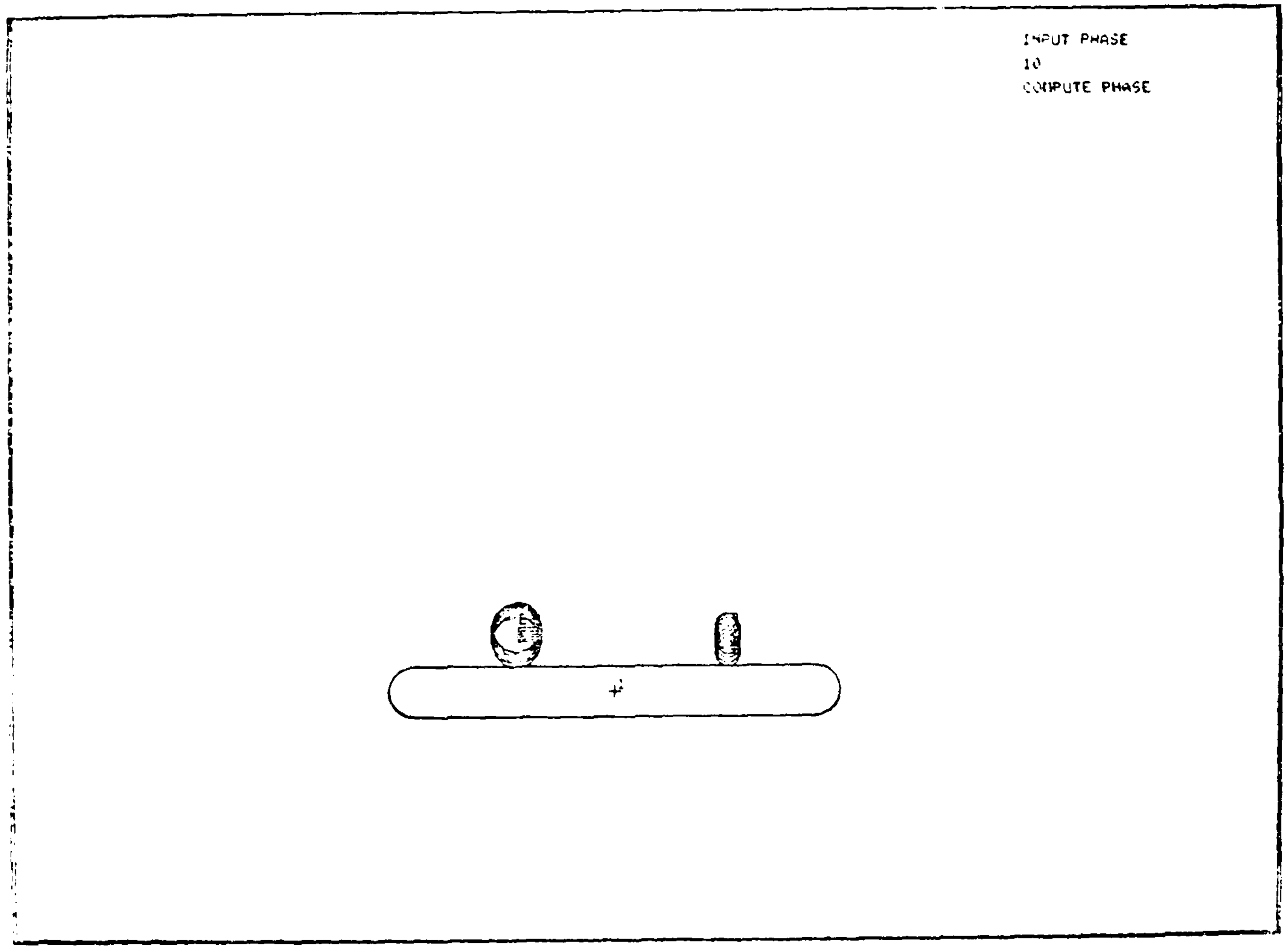
a.

Figure 11. Example No. 1 (Sheet 1 of 10)



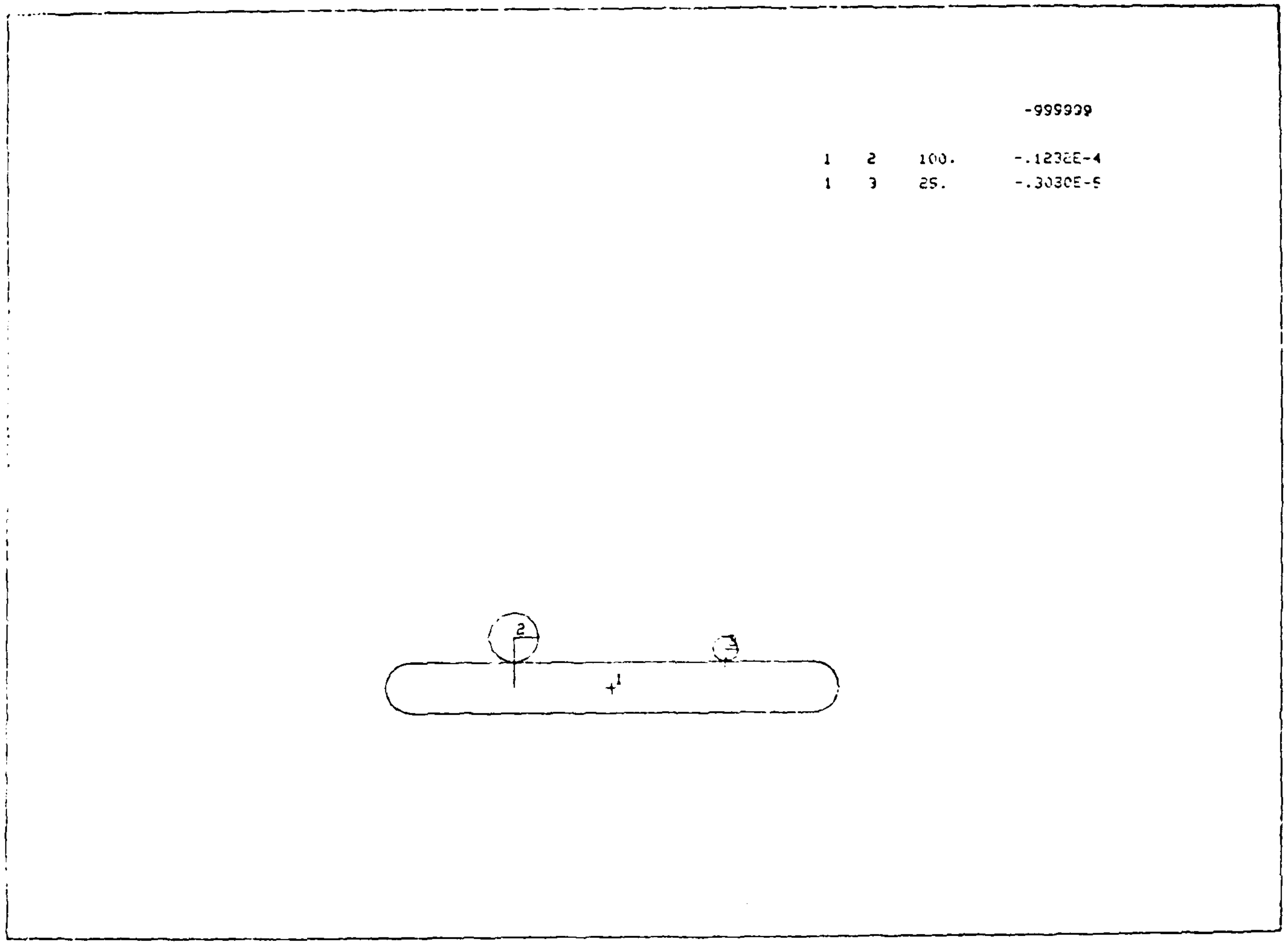
b.

Figure 11. (Sheet 2 of 10)



c.

Figure 11. (Sheet 3 of 10)



d.

Figure 11. (Sheet 4 of 10)

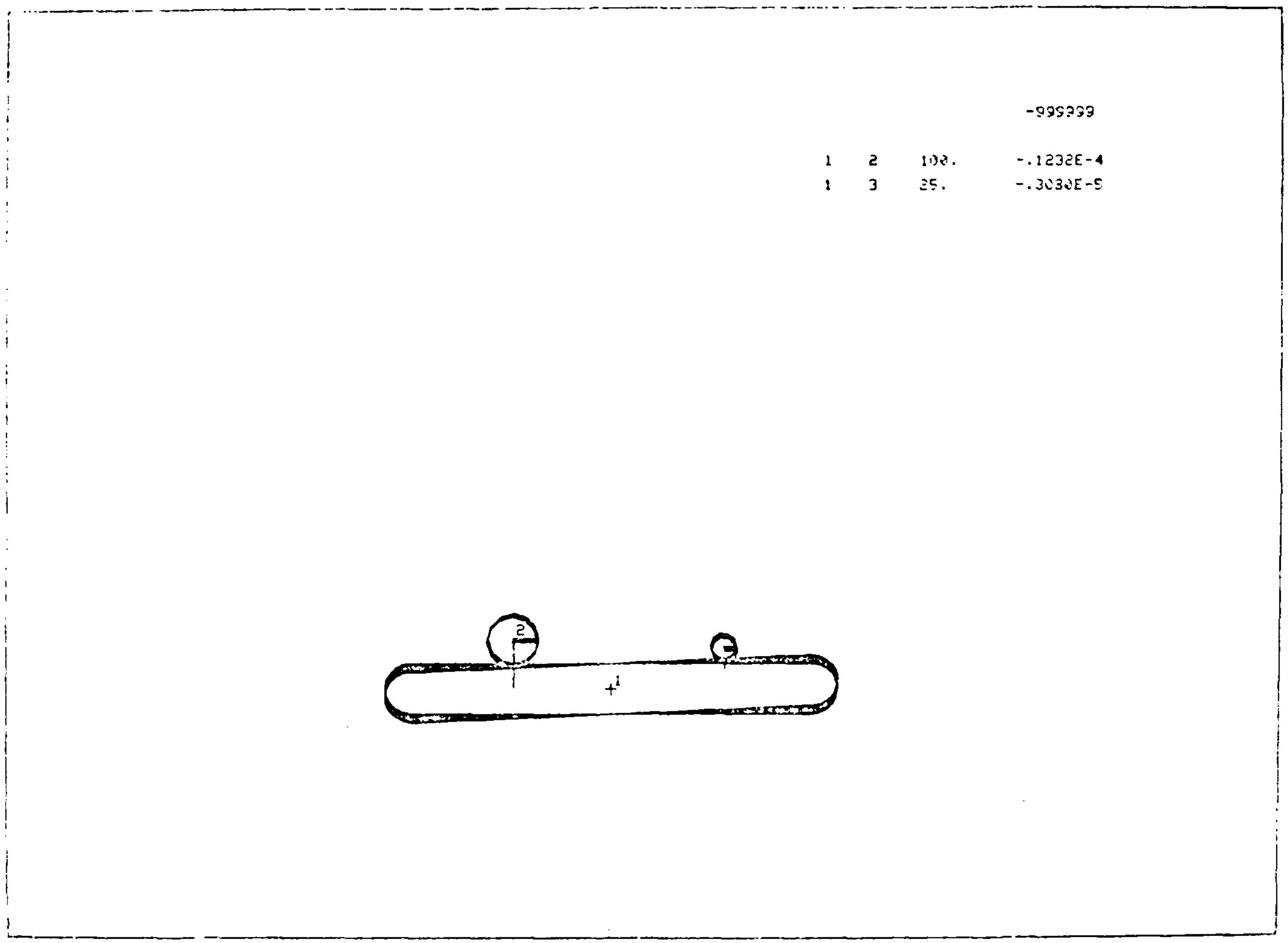
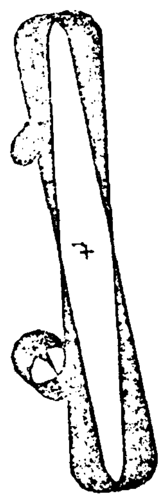


Figure [illegible]. (Sheet 5 of 10)



f.

Figure 11. (Sheet 6 of 10)

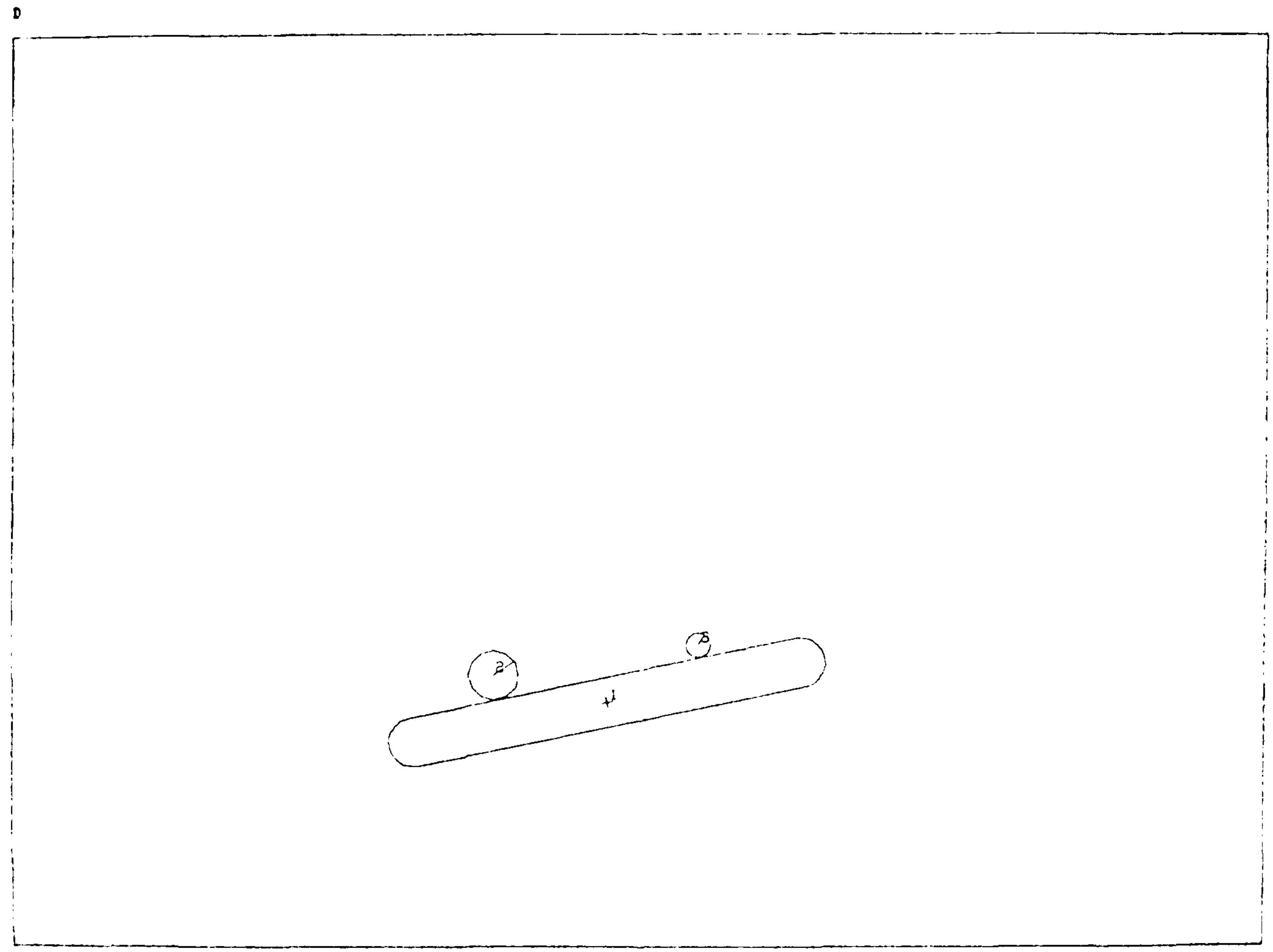


Figure 11. Sheet [illegible]

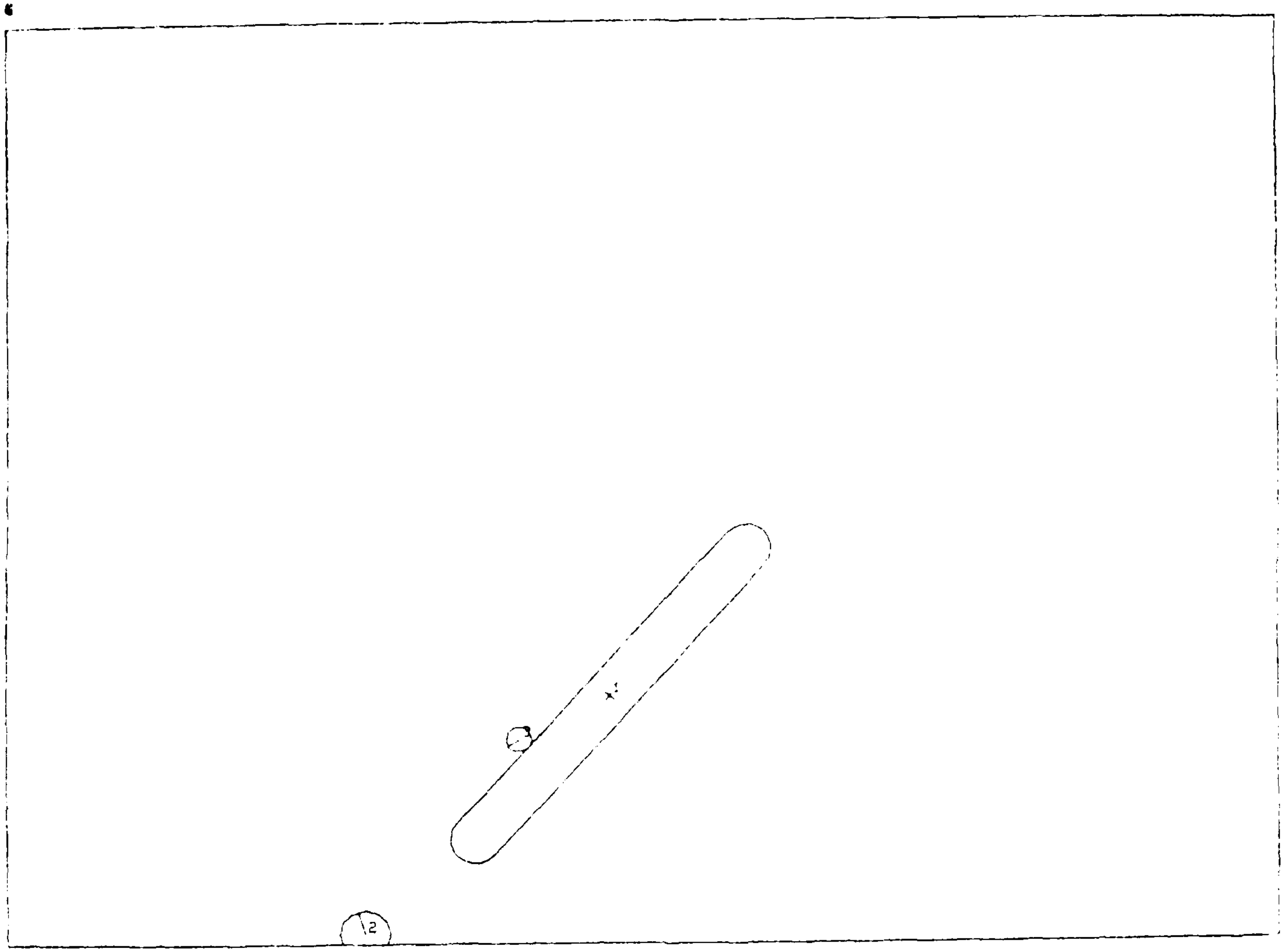


Figure 11. (Sheet 5 of 10)

a. The bar element was created by positioning the crosshair cursor at the left end of desired location of that element, and the key "1" was struck. The crosshair cursor reappeared. With the thumbwheel, the crosshair cursor was moved to the right end of the bar element, and the key "2" was struck. The bar element was then drawn.

b. The larger disc element was then created by positioning the crosshair cursor at the desired location, and the "spacebar" was struck. A disc with a radius of 20 screen units (the default value) was then drawn at the location shown in Figure 11a.

c. The smaller disc element was created by first hitting the "R" key and entering the parameter "10." This signals the computer to create any further elements with a radius of 10 screen units (see upper right corner of Figure 11a). The crosshair cursor was placed at the location of the center of the small disc and the "space bar" struck.

88. After creating the desired elements, it is possible to enter the compute phase of the program by striking key "C." Immediately, the message "COMPUTE PHASE" appears on the screen (Figure 11b). At this point, the command "R" was given and the elements were redrawn and numbered. Following this step, the bar element (element 1) was restrained from all movement by positioning the crosshair cursor at the center of element 1 and striking the key "[illegible]."

89. Striking the "G" key caused the program to compute a total of 250 time steps of motion. As elements move, their positions are drawn on the screen. The Tektronix terminal has a "memory" screen; that is, once something is drawn on the screen, it is retained until the entire screen is caused to be erased. Figure 11c shows the intermediate motions of the two discs as they fall under gravity to the fixed bar element.

90. After erasing the screen (key "[illegible]" or "[illegible]"), the current position of the elements was redrawn by striking the "D" key as shown in Figure 11d. [illegible], at this point, the [illegible] was given a large tensile contact strength [illegible] crosshair cursor over the small disc [illegible] striking key "T" [illegible] value for tensile [illegible]

this time on, it is not possible to cause the small disc element 3 to separate from the bar element. The forces (in both normal and shear directions) are outputted to the upper right corner of the screen by the command "O" followed by "H." Thus, for the contact between elements 1 and 2, the normal force is 100.0 and the shear force very small (-0.1234×10^{-4}). The normal force between elements 1 and 3 is 25.0, and the shear force is -0.308×10^{-5}. Recalling that the weight of the "standard" disc (radius = 20 screen units) is 100.0, the computed forces are correct. (Element 3 has a radius of 10 screen units and would weigh only one-fourth the amount of the "standard" disc.)

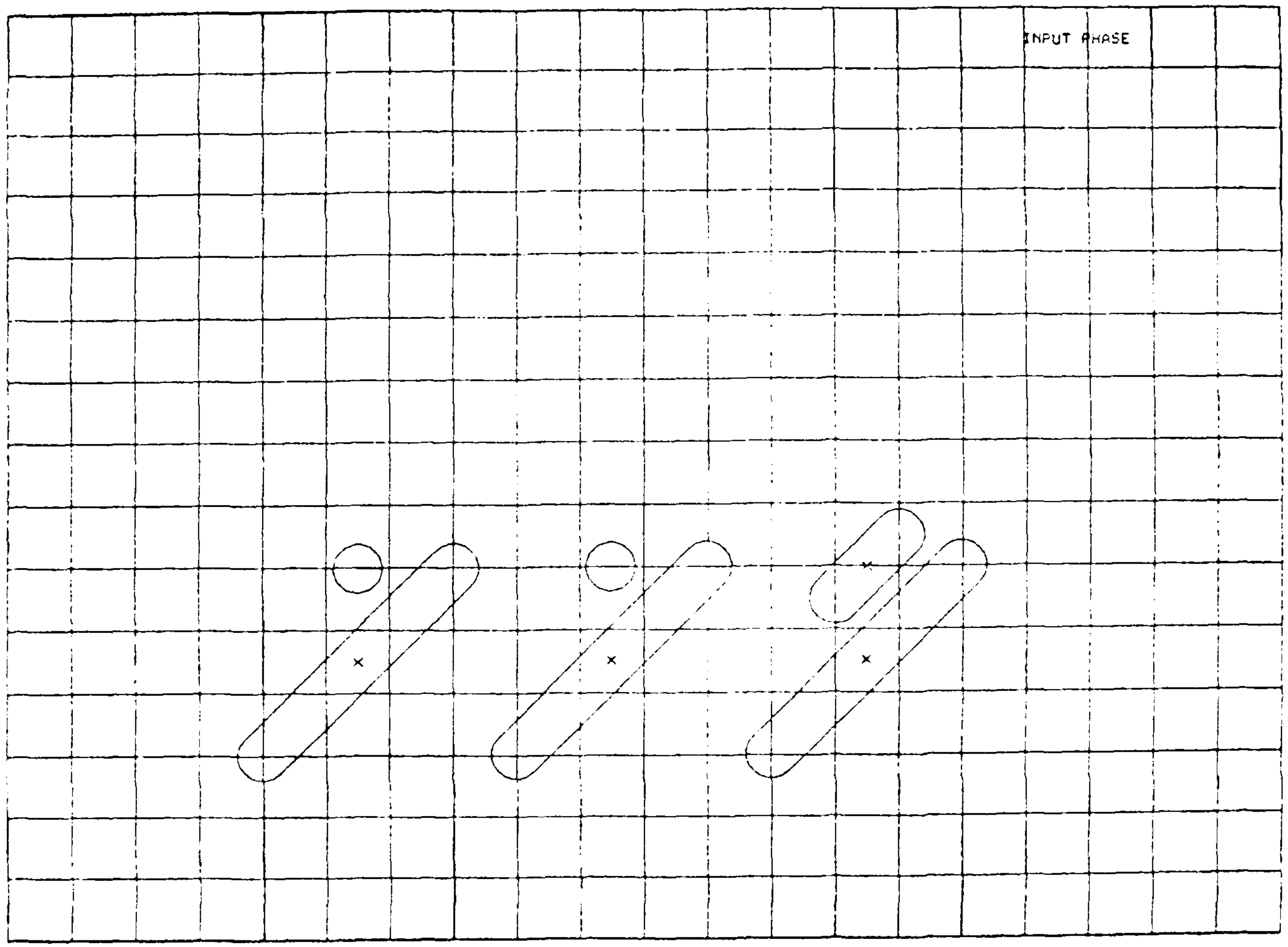
91. At this point, the bar element is released from its fixity in all directions through the command "U" (with the crosshairs centered on the bar element). This command is followed by restraining the bar element from movement in both the horizontal and vertical directions by striking "X" followed by "Y" (again with the crosshair cursor on element 1). The bar element is now prevented from translating but is free to rotate about its center of gravity. To again begin calculations of motion, the "G" key is struck. Since the large disc 2 exerts a counterclockwise moment on the bar, the system begins to rotate as shown in Figure 11e.

92. Repeated "G" commands lead to the situation shown in Figure 11f and Figure 11g. The bar continues to rotate, and the disc elements roll down the plane. The lines joining the disc element centers to the periphery of the discs indicate the amount of rotation experienced by the discs (Figure 11g). At a later time (Figure 11h), the larger disc 2 has completely rolled off the bar element and will soon disappear from the bottom of the screen.

93. Recalling that, at an earlier time, it was specified that disc 3 should have a large tensile contact "strength," it should be expected that element 3 could not roll off the bar. Figure 11i indicates exactly that situation. Disc 3 rolls around to the underside of the bar element. Figure 11j is a refreshed plot of the final element positions occurring before calculations were terminated.

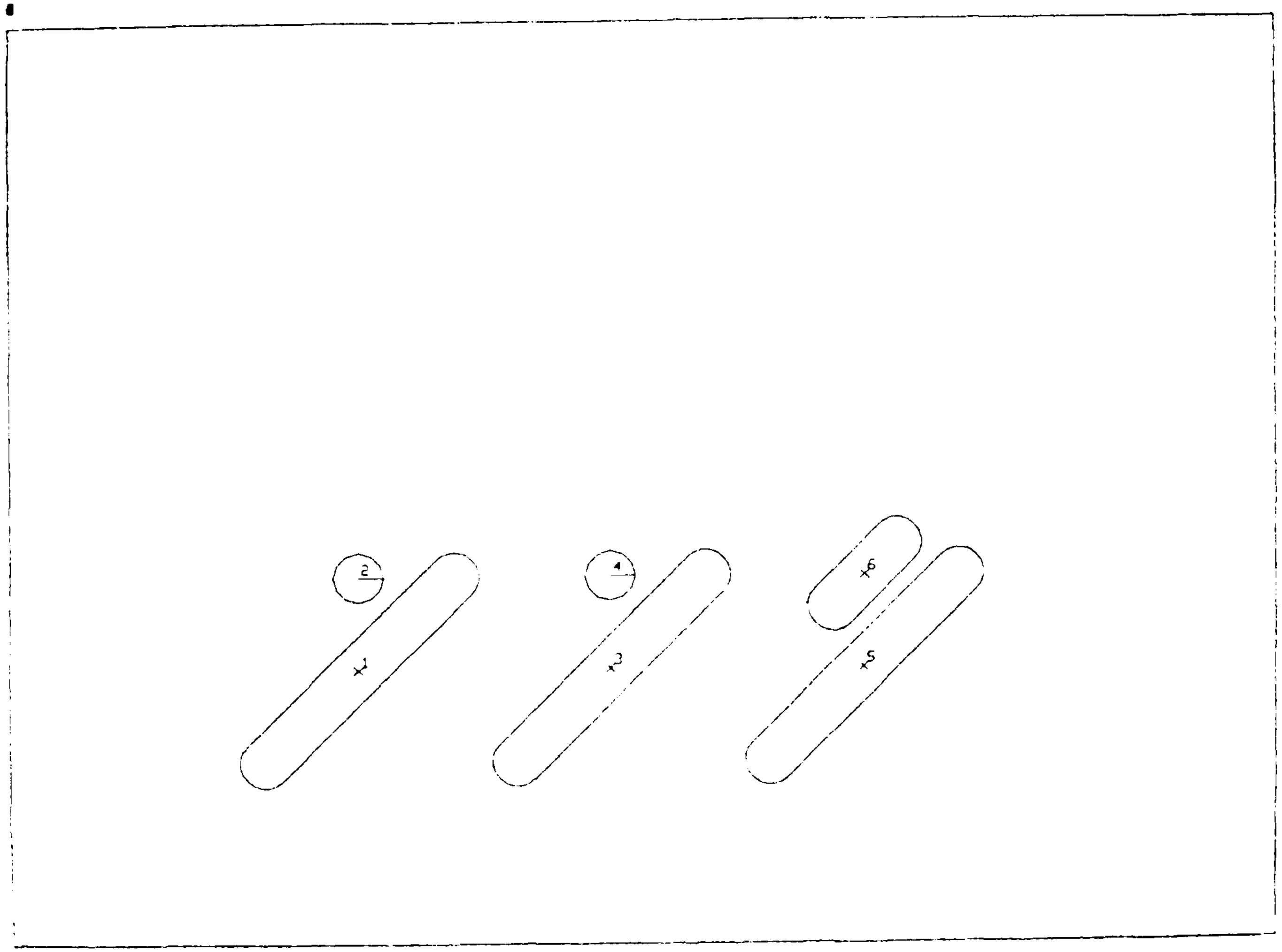
94. As a second simple example, consider the results given in Figure 12. In this example, three problems are being solved simultaneously. As soon as the input phase is entered, the key "4" is struck and causes a grid to be drawn on the screen. The grid is helpful to orient the three long bar elements at a 45-deg angle (Figure 12a). After inputting the three bar elements and the two discs, the compute phase is called and the elements are numbered (Figure 12b). All of the lower bar elements are then fixed by typing "F" in sequence for those elements. Then, element 4 is restrained from rotating by placing the crosshair cursor at its center and striking "M." After typing "G," the upper elements fall and proceed to slide down (element 2 rolls down) the long bar elements (Figure 12c). The default value for coefficient of friction is 0.5; thus the shear forces should never exceed one half of the normal forces. Figure 12c gives the normal and shear forces at all contacts. In every instance, the shear force is one half of the normal force. Notice that there are two 5-6 contacts, the first one being a larger value than the second. This is predictable since most of the weight should be felt by the lower contact.

95. The third example is illustrated in Figure 13. The first figure of this series shows a situation in which disc elements 6 and 7 are surrounded by a boundary of fixed bar elements. There is, however, an opening between elements 2 and 3. After describing the locations of the elements in the input phase, the compute phase is entered. Elements 1 through 5 are immediately fixed through the use of the "F" key. Following that step (Figure 13a), the weights of all elements are set to zero by the use of the "W" key (the masses remain unchanged). Then, the damping constant is reduced to zero by using the "C" sequence. Next, the coefficient of friction is set to zero (using the "Ø" key) so that the angles of reflection will equal the angles of incidence in whatever motions ensue. Through the use of the "V" key, the initial velocity of element 6 is set to v_x = -100 screen units per second, v_y = -100, and the angular velocity is set to zero. For element 7, the "V" key was used to set v_x = 100 units per second and v_y = 100 units. Figure 13b



a.

Figure [illegible]. Example No. 2 (Sheet 1 of [illegible])



b.

Figure 12. (Sheet 2 of 3)

1	2	70.95	-35.47
3	4	70.71	-35.36
5	6	96.71	-48.36
5	6	54.39	-27.19

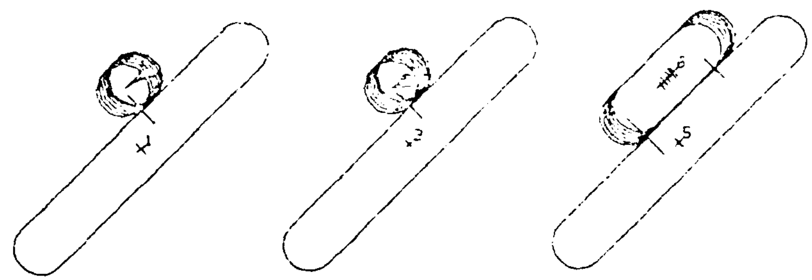
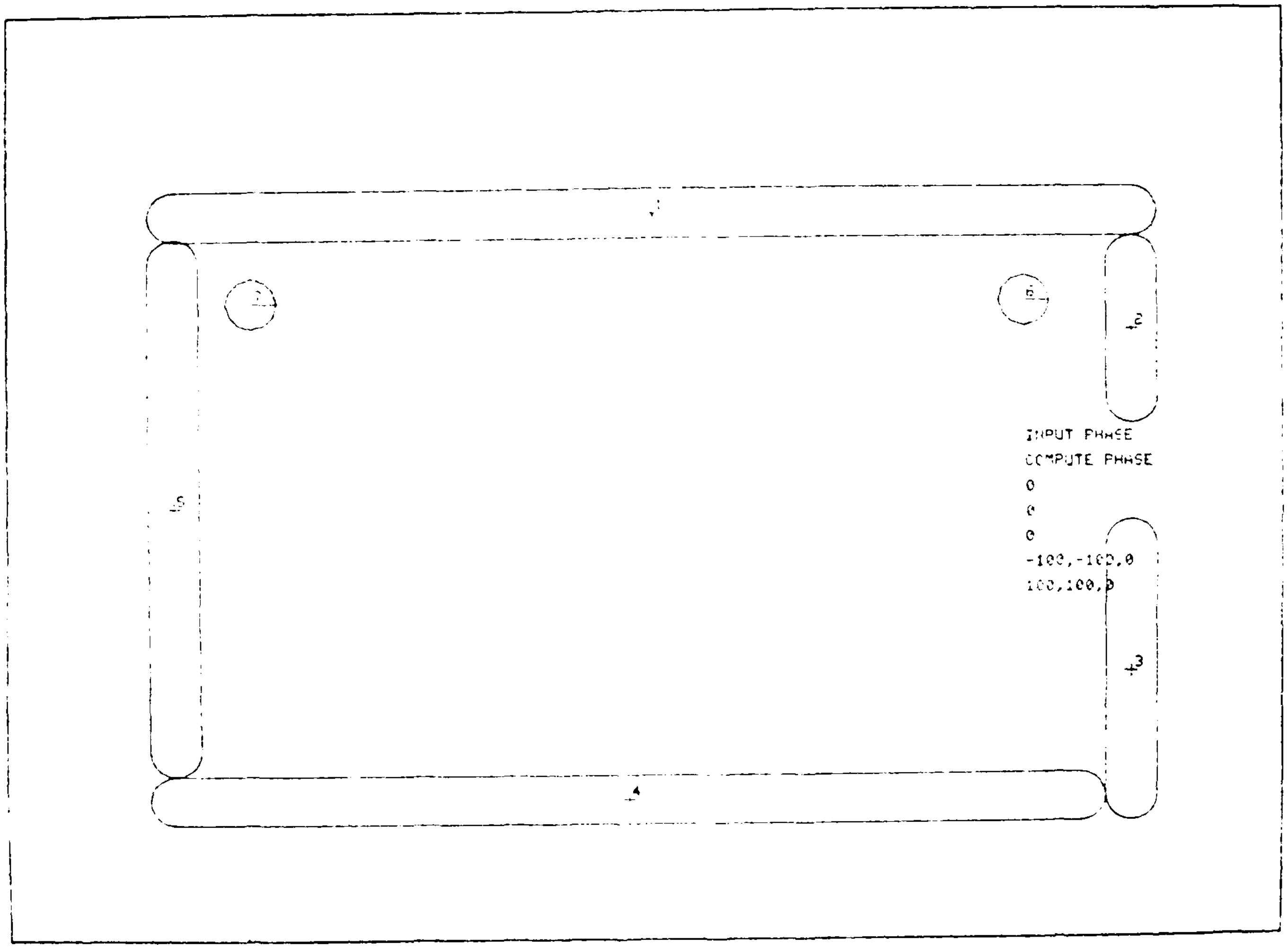
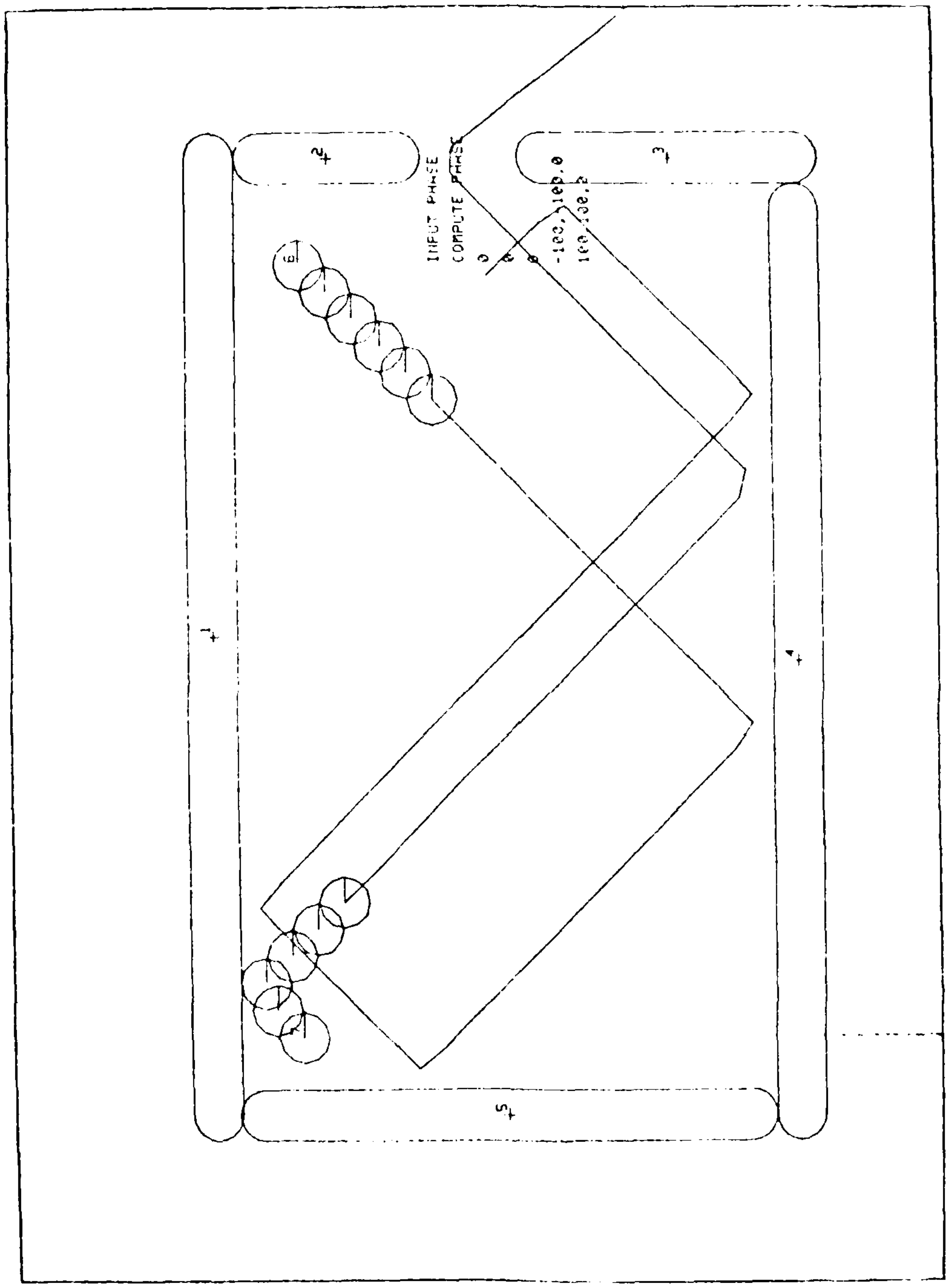
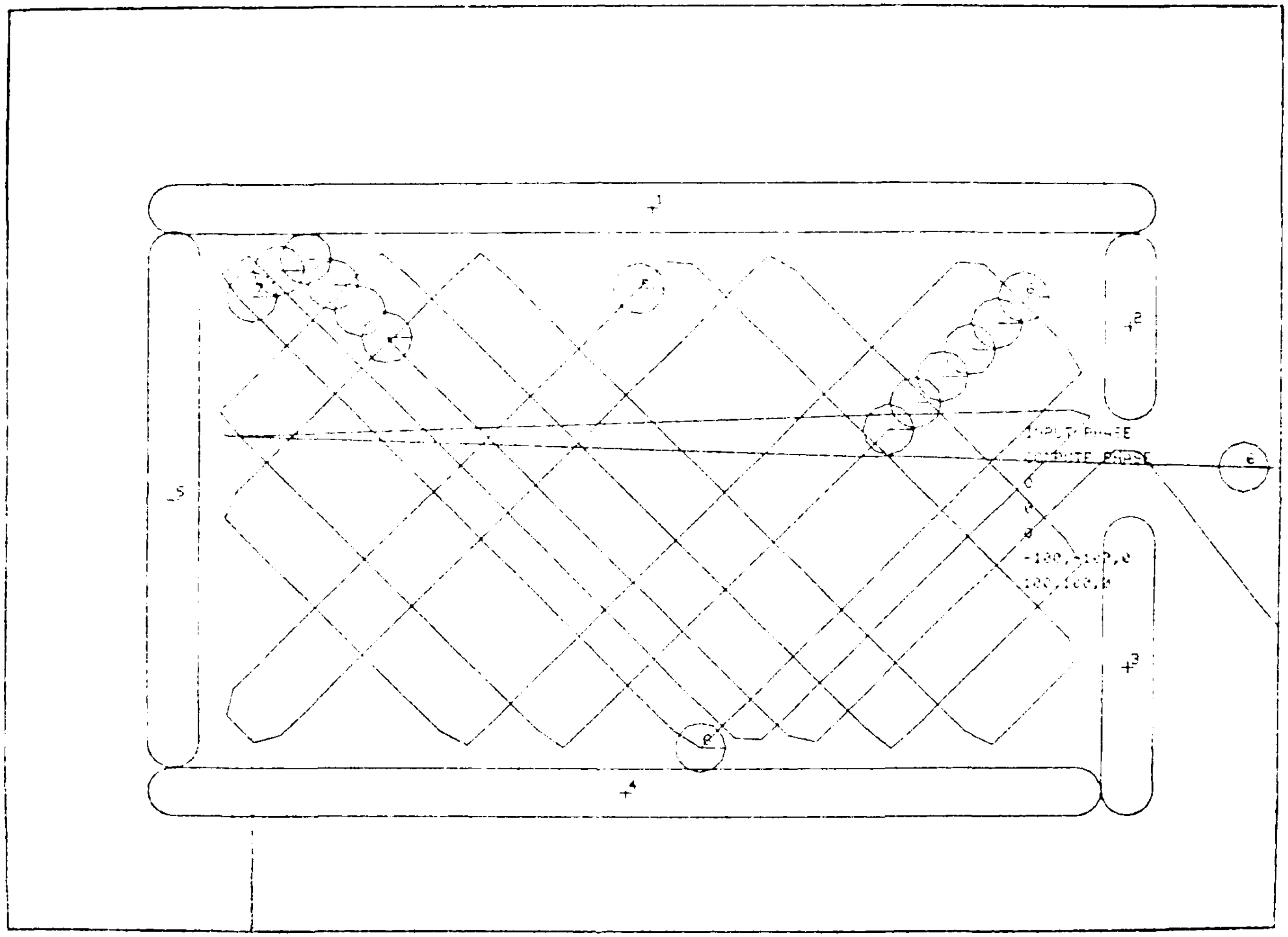


Figure 12. (Sheet 3 of 3)



[illegible] (Sheet 1 of 3)


INPUT PHASE
COMPUTE PHASE
1
2
3
4
5



c.

Figure 15. (Sheet 3 of 3)

shows the resulting motions after striking key "G." Notice that disc 6 proceeds downward to the left. Disc 7 starts upward to the right, strikes the upper boundary, and turns downward to the right. Following the first calculation cycle of 250 time steps, output is converted to the "vector" mode by hitting key "*." When in the vector mode, the elements are no longer drawn. The output now consists only of the movement of the element centers. Element 7 proceeds to rebound off the lower boundary, then strikes the lower end of element 2, and exits through the opening (Figure 13b). Element 6, however, does not quickly find an exit through the opening. In Figure 13c, element 6 suffers a total of 21 rebounds before hitting the rounded portion of element 2. Element 6 then is deflected in an almost horizontal direction to the left wall and shoots across and out through the opening (almost dead center).

96. The purpose of the three example problems is to demonstrate the flexibility and range of this program. Problems involving quasi-static behavior and large rapid motions are included within the same framework. Indeed, at one time, this program contained options for including "inverse square of the distance" forces so that the user could compute orbits of gravitational bodies. These options are not germane to the general fields of geotechnical engineering and, as such, are not present in this version. However, the program DISC is basically a tool that will solve many fundamental problems in rigid body mechanics.

PART V: CONCLUSIONS AND RECOMMENDATIONS

Conclusions

97. A computer program entitled DISC has been developed for analyzing disc (and bar) shaped elements. The formulations, based upon the distinct element method, allow for large, rapid movements, or quasi-static movements. The discretized behavior of the elements are kinematically calculated by the program DISC so that individual system particles (the discs) are capable of rolling, sliding, displacing each other, etc. The distinct element method provides a contrast to common continuum analysis procedures, such as the finite element method.

98. At the beginning of this study, the writer was optimistic that this method would be directly applicable to problems encountered in geotechnical engineering. During the course of this study, many "computer runs" were made in the hope of modeling a variety of common geotechnical phenomena, such as slope stability, retaining walls, bearing capacity, and pile driving. None of these attempts were satisfactory to the extent required for reporting in this document. That is, the writer has not been able to extend the method to common geotechnical problems in a direct fashion. However, many geotechnical problems can be physically modeled by an assemblage of discs. The results of such modeling could lead to (at the present) an understanding of the details of the individual particle motion. For example, a retaining wall could be physically modeled by an assemblage of discs, as could a projectile penetration problem. The program DISC is an extremely powerful model to analyze disc-shaped objects. To this purpose, the writer knows of no other method that is nearly as accurate or flexible. For example, a paper was recently published describing a program quite similar to DISC (Cundall and Strack, 1979). In this paper, the comparison was made between a computer modeling program entitled BALL (similar to DISC) and photoelastic studies of physical discs. The comparison was excellent; however, once again, both studies were modeling discs.

99. It shall remain for others to extend the distinct element method to more meaningful situations. It is the writer's opinion that this type of analysis will eventually be adopted in many areas of engineering. The concept of analyzing discrete particles in an explicit scheme is attractive to the formulations of geotechnical problems.

Recommendations

100. It must be emphasized that the computer program DISC was written to be quite general. If a subsequent user of this program has to analyze a particular problem, he should make an attempt to modify the basic, general program to his needs. For example, if the problem requires only a quasi-static analysis, many savings could be accomplished by fewer searches for near contacts. Researchers generally tend to direct new concepts to a large audience and always try to account for any possible application of their developments. A user of this research would be well advised to simply store a copy of the original program DISC within his files and then modify and enhance whichever portions of DISC that fit his applications. That is, adapt the program to the particular need.

REFERENCES

Cundall, P. A. 1971. "A Computer Model for Simulating Progressive Large Scale Movements in Blocky Rock Systems," Paper II-8, Proceedings of Symposium on Rock Fractive, International Society for Rock Mechanics, Nancy, France.

Cundall, P. A. 1974. "A Computer Model for Rock Mass Behavior Using Interactive Graphics," Technical Report 2-74, U. S. Army Corps of Engineers, Missouri River Division, Omaha, Nebr.

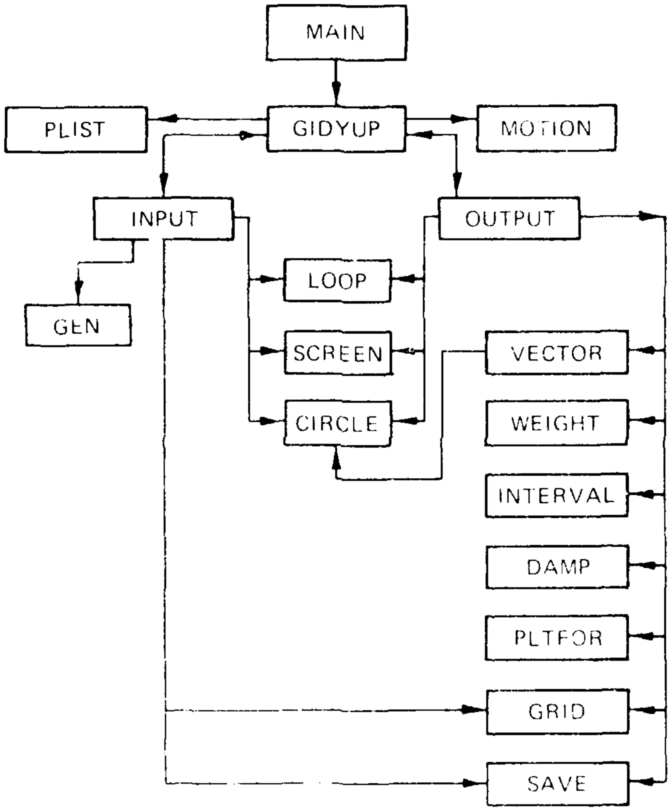
Cundall, P. A., and Strack, O. D. L. 1979. "A Discrete Numerical Model for Granular Assemblies," Geotechnique, Vol 29, No. 1, pp 47-65.

Voegele, M. D. 1979. "Rational Design of Tunnel Supports: An Interactive Graphics Based Analysis of the Support Requirements of Excavations in Jointed Rock Masses," Technical Report GL-79-15, U. S. Army Engineer Waterways Experiment Station, CE, Vicksburg, Miss.

West Point Military Academy and U. S. Army Engineer Waterways Experiment Station. 1975 (revised 1979). "Primer on Computer Graphics Programming," U. S. Army Engineer Waterways Experiment Station, CE, Vicksburg, Miss.

APPENDIX A: SCHEMATIC DIAGRAM OF DISC AND FORTRAN SOURCE LISTING

SCHEMATIC FLOW OF DISC



```
C    This is the computer program DISC.
C     This program was written by John B. Palmerton, U.S. Army
C     Corps of Engineers Waterways Experiment Station,
C     P.O. Box 631, Vicksburg,MS.  Telephone 601-634-3357
C     FTS 542-3357
      PARAMETER M=50
      PARAMETER MSZ=M*24
      PARAMETER MLIST=M*6
      COMMON/NUMS/NWO,NWL
      COMMON/JUNK/N,SKN,DKN,TIME,DT,WF,DF
      COMMON/LIST/LL(MLIST)
      COMMON/STORE/AA(MSZ),NA(M),NAA(M)
      COMMON/FORCE/FXSUM(M),FYSUM(M),FMSUM(M),FX(M),FY(M),FM(M)
      COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
      COMMON/VELO/UX(M),UY(M),UT(M),VX(M),VY(M),VT(M)
      COMMON/INIT/XIN(M),YIN(M),TIN(M)
      COMMON/PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
      COMMON/GRAV/AGRAV
      HS=10.
      VS=13.
      CALL USTART
      CALL USET('RASTER')
      CALL UWINDO(0.,1000.,0.,750.)
      CALL UDAREA(0.,1000.,0.,750.)
      CALL UPSET('HORIZONTAL',HS)
      CALL UPSET('VERTICAL',VS)
      AGRAV=32.2
      CALL GIDYUP(NFIRST)
      STOP
      END
      SUBROUTINE GIDYUP(NFIRST)
      PARAMETER M=50
      COMMON/NUMS/NWO,NWL
      COMMON/JUNK/N,SKN,DKN,TIME,DT,WF,DF
      COMMON/FORCE/FXSUM(M),FYSUM(M),FMSUM(M),FX(M),FY(M),FM(M)
      COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
      COMMON/INIT/XIN(M),YIN(M),TIN(M)
      COMMON/PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
      CALL USET('SYSTEMAXIS')
      CALL UWINDO(0.,1000.,0.,750.)
      CALL UMOVE(0.,0.)
      CALL UROTAT(0.)
      NFIRST=0
      INT=1
      CALL INPUT(NFIRST,INT)
      TIME=0.0
      NW=N
      KOUT=50
      KK=250
      CALL OUTPUT(KK,0,NFIRST,NW,LEN,INT)
      CALL VECTOR(N,1,2
      CALL PLIST(LEN)
      KASK=KK
      NWL=0
   50 CONTINUE
      NW=INT
      CALL MOTION(NW)
      KASK=KASK-50
      KOUT=KOUT-50
      IF(KOUT.GT.0) GO TO 75
      CALL OUTPUT(KK,1,NFIRST,NW,LEN,INT
      CALL PLIST(LEN)
      KOUT=50
   75 CONTINUE
      NFIRST=1
      IF(KASK.GT.0) GO TO 150
      CALL OUTPUT(KK,0,NFIRST,NW,LEN,INT)
      CALL PLIST(LEN)
      KASK=KK
  150 CONTINUE
      IF(2.EQ.2) GO TO 50
      STOP
      END
      SUBROUTINE INPUT(NFIRST,INT)
      PARAMETER M=50
      PARAMETER MSZ=M*24
      COMMON/NUMS/NWO,NW
      COMMON/JUNK/N,SKN,DKN,TIME,DT,WF,DF
      COMMON/STORE/AA(MSZ),NA(M),NAA(M)
      COMMON/FORCE/FXSUM(M),FYSUM(M),FMSUM(M),FX(M),FY(M),FM(M)
      COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
      COMMON/VELO/UX(M),UY(M),UT(M),VX(M),VY(M),VT(M)
      COMMON/INIT/XIN(M),YIN(M),TIN(M)
      COMMON/PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
      COMMON/GRAV/AGRAV
      DATA XMIN,XMAX,YMIN,YMAX/0.,1000.,0.,750./
      CHARACTER IP*1
      DIMENSION COORD(2)
      DIMENSION VEL(3)
      CHARACTER JP*1
      CALL UWINDO(0.,1000.,0.,750.)
      CALL USET('TEXT')
      CALL UERASE
      CALL UPAUSE
      CALL UOUTLN
      CALL SCREEN(XP,YP)
      CALL UPRINT(XP,YP,'INPUT PHASE')
      CALL UOUTLN
      DO 4 I=1,M
      UX(I)=0.0
      UY(I)=0.0
    4 UT(I)=0.0
      DO 6 I=1,MSZ
    6 AA(I)=0.0
      IF(NFIRST.NE.0) GO TO 11
      DO 5 I=1,M
      X(I)=0.
      Y(I)=0.
      T(I)=0.
      R(I)=20.
      S(I)=0.0
      W(I)=0.0
      FRC(I)=0.5
      TEN(I)=0.0
      IFIX(I)=0
      WGTF(I)=1.
      FX(I)=0.0
      FY(I)=0.0
      FM(I)=0.0
      UX(I)=0.0
      UY(I)=0.0
      UT(I)=0.0
    5 CONTINUE
      NFIRST=0
      MAX=M
      N=1
```

```
11 RR=20.
10 CALL UBELL
CALL USET('TEXT')
CALL UGRIN(X1,Y1,IP)
IF(IP.EQ.'g') GO TO 700
IF(IP.EQ.'R') CALL GIDYUP(NFIRST)
IF(IP.EQ.';'.OR.IP.EQ.'+') CALL GRID(XMIN,XMAX,YMIN,YMAX)
IF(IP.EQ.'(') GO TO 10
IF(IP.EQ.') CALL SAVE(N,0)
IF(IP.EQ.') GO TO 51
IF(IP.EQ.'I') GO TO 200
IF(IP.EQ.'1') GO TO 100
IF(IP.EQ.'2') GO TO 101
IF(IP.EQ.'R') GO TO 180
IF(IP.EQ.'J') GO TO 20
IF(IP.EQ.'P') GO TO 20
IF(IP.EQ.'D') GO TO 50
IF(IP.EQ.'C') GO TO 40
IF(IP.EQ.'B') CALL GEN(N,RR,X1,Y1)
IF(IP.EQ.'B') GO TO 10
IF(IP.EQ.'E') GO TO 500
IF(IP.EQ.'A') GO TO 14
IF(IP.NE.' ') CALL SCREEN(XP,YP)
IF(IP.NE.' ') CALL UPRINT(XP,YP,'????\')
IF(IP.NE.' ') GO TO 10
8 X(N)=X1
Y(N)=Y1
R(N)=RR
CALL CIRCLE(X(N),Y(N),R(N),S(N),T(N))
N=N+1
IF(N.GT.MAX) RETURN
GO TO 10
14 D1=X1-X(N-1)
D2=Y1-Y(N-1)
D3=SQRT(4.*RR*RR/(D1*D1+D2*D2))
X1=X(N-1)+D1*D3
Y1=Y(N-1)+D2*D3
S(N)=S(N-1)
T(N)=T(N-1)
GO TO 8
40 CALL LOOP(X1,Y1,J,N)
IF(J.GT.N) GO TO 10
CALL USET('XYCOOR')
CALL SCREEN(XP,YP)
COORD(1)=X1
COORD(2)=Y1
CALL UPRINT(XP,YP,COORD)
CALL USET('TEXT')
GO TO 10
20 CALL LOOP(X1,Y1,J,N)
VEL(3)=0.0
IF(J.GT.N) GO TO 10
CALL USET('REAL')
CALL SCREEN(XP,YP)
CALL UREAD(XP,YP,VEL,3.,FL)
IF(VEL(1).LT.0.01) VEL(1)=X(J)
IF(VEL(2).LT.0.01) VEL(2)=Y(J)
VEL(3)=VEL(3)/57.2957795
T(J)=T(J)+VEL(3)
IF(IP.EQ.'J') GO TO 27
DO 26 I=1,N-1
IF(I.EQ.J) GO TO 26
B=ATAN2(Y(I)-Y(J),X(I)-X(J))
B=B+VEL(3)
D=SQRT((X(I)-X(J))**2+(Y(I)-Y(J))**2)
X(I)=X(J)+D*COS(B)
Y(I)=Y(J)+D*SIN(B)
T(I)=T(I)+VEL(3)
26 CONTINUE
27 CONTINUE
X(J)=VEL(1)
Y(J)=VEL(2)
IF(IP.EQ.'J') GO TO 10
XINT=X(J)-X1
YINT=Y(J)-Y1
DO 30 I=1,N-1
IF(I.EQ.J) GO TO 30
X(I)=X(I)+XINT
Y(I)=Y(I)+YINT
30 CONTINUE
50 CALL USET('TEXT')
CALL UERASE
CALL UPAUSE
51 CONTINUE
DO 55 I=1,N-1
CALL CIRCLE(X(I),Y(I),R(I),S(I),T(I))
55 CONTINUE
GO TO 10
180 CALL USET('REAL')
CALL SCREEN(XP,YP)
CALL UREAD(XP,YP,RR,1.,FL)
GO TO 10
100 CALL UGRIN(X2,Y2,JP)
101 CONTINUE
IF(JP.EQ.IP) GO TO 110
X(N)=(X2+X1)/2.
Y(N)=(Y2+Y1)/2.
T(N)=ATAN2(Y2-Y1,X2-X1)
S(N)=0.5*SQRT((X2-X1)**2+(Y2-Y1)**2)
R(N)=RR
GO TO 150
110 CONTINUE
T(N)=ATAN2(Y1-Y2,X1-X2)
XP=2.*RR*COS(T(N))+X2
YP=2.*RR*SIN(T(N))+Y2
X(N)=(X1+XP)/2.
Y(N)=(Y1+YP)/2.
S(N)=0.5*SQRT((XP-X1)**2+(YP-Y1)**2)
R(N)=RR
X2=X1
Y2=Y1
150 CALL CIRCLE(X(N),Y(N),RR,S(N),T(N))
N=N+1
GO TO 10
500 WF=1.0
CALL USET('TEXT')
CALL SCREEN(XP,YP)
CALL UPRINT(XP,YP,'COMPUTE PHASE\')
DF=1.0
N=N-1
CALL SAVE(N,1)
DO 25 I=1,N
XIN(I)=X(I)
YIN(I)=Y(I)
TIN(I)=T(I)
W(I)=(R(I)*R(I)*3.1415926+R(I)*S(I)*2.)
& /12.5663706
25 XM(I)=W(I)/AGRAV
```

```
IF(NFIRST.NE.0) RETURN
XMLESS=99999999.
UMAX=0.000001
XN2=1.
DO 35 I=1,N
XN2=XN2+1
IF(XM(I).LT.XMLESS) XMLESS=XM(I)
IF(U(I).GT.UMAX) UMAX=U(I)
35 CONTINUE
SKN=SQRT(XN2)*UMAX
SKN=10.*SKN
DT=0.05*SQRT(XMLESS/SKN)
DT=2.*DT
DO 210 I=1,N
210 NAA(I)=I
NUO=N
NU=2*N
RETURN
200 CALL USET('REAL')
CALL SCREEN(XP,YP)
CALL UREAD(XP,YP,CY,1.,FL)
INT=CY+0.1
IF(INT.GT.50) INT=50
GO TO 10
700 CALL SCREEN(XP,YP)
CALL USET('REAL')
CALL UREAD(XP,YP,AGRAU,1.,FL)
GO TO 10
END
SUBROUTINE OUTPUT(KASK,NN,NFIRST,NU,LEN,INT
PARAMETER MC=42
PARAMETER M=50
PARAMETER MSZ=M*24
COMMON/NUMS/NUO,NUE
COMMON/JUNK/N,SKN,DKN,TIME,DT,UF,DF
COMMON/STORE/AA(MSZ),NA(M),NAA(M)
COMMON/FORCE/FXSUM(M),FYSUM(M),FMSUM(M),FX(M),FY(M),FM(M)
COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),U(M),XM(M)
COMMON/VELO/UX(M),UY(M),UT(M),VX(M),VY(M),VT(M)
COMMON/INIT/XIN(M),YIN(M),TIN(M)
COMMON/PARA/IFIX(M),UGTF(M),FRC(M),TEN(M)
CHARACTER ICHAR*1(MC)/
& '1','!','N','S','K','Q','I','Z',' ',' ','a',
& 'b','!','d','c','*','a',
& '$','4','D','E','/','+','G','0',' ','T',' ','0',' ',
& '?','F','X','V',' ',' ',' ',' ',' ',' ',' ','r'/
CHARACTER IP*1
DIMENSION UEL(3),UL(2)
DATA XMIN,XMAX,YMIN,YMAX/0.,1000.,0.,750./
DATA XMN,XMX,YMN,YMX/0.,1000.,3.,750./
DATA MJ,XMJ,YMJ/0.,0.,0./
CALL UWINDO(XMIN,XMAX,YMIN,YMAX)
52 IP=' '
IF(NFIRST.NE.0) GO TO 55
JP=1
KP=1
56 CALL USET('TEXT')
55 IF(NN.EQ.1) GO TO 2
100 CALL USET('TEXT')
IP=' '
CALL USET('SYSTEMAXIS')
IFND=0
DO 110 I=1,MC
IF(IP.EQ.ICHAR(I)) IFND=1
110 CONTINUE
IF(IFND.EQ.1) GO TO 120
CALL SCREEN(XP,YP)
CALL UPRINT(XP,YP,'????')
CALL UBELL
GO TO 130
120 CONTINUE
CALL USET('HARDWARE')
CALL UWINDO(XMIN,XMAX,YMIN,YMAX)
IM=XMX+YMX+0.1
IF(IM.NE.1750) CALL USET('SOFTWARE')
IF(NN.EQ.1) RETURN
CALL USET('TEXT')
130 CALL USET('USERAXIS')
CALL UGRIN(P1,P2,IP)
CALL USET('SYSTEMAXIS')
IF(IP.EQ.'1'.OR.IP.EQ.'!') JP=-JP
IF(IP.EQ.'C') GO TO 11
IF(IP.EQ.'N') GO TO 1320
IF(IP.EQ.'S') CALL SAVE(N,I)
IF(IP.EQ.'K') CALL UERASE
IF(IP.EQ.'Q') CALL DAMP(IF
IF(IP.EQ.'I') CALL INTERAL(KASK
IF(IP.EQ.'Z') GO TO 200
IF(IP.EQ.'$'.OR.IP.EQ.'4') KP=-KP
IF(IP.EQ.'D'.OR.IP.EQ.'d') GO TO 2
IF(IP.EQ.'E') N=N+1
IF(IP.EQ.'E') NFIRST=1
IF(IP.EQ.'E') CALL INPUT(NFIRST,INT)
IF(IP.EQ.'E') GO TO 56
IF(IP.EQ.'/'.OR.IP.EQ.'+') CALL GRID(XMIN,XMAX,YMIN,YMAX)
IF(IP.EQ.'?') GO TO 350
IF(IP.EQ.'m') GO TO 1700
IF(IP.EQ.'u') GO TO 1200
IF(IP.EQ.'r') GO TO 1300
IF(IP.EQ.'a') GO TO 1400
IF(IP.EQ.'b') GO TO 1350
IF(IP.EQ.'a') GO TO 1800
IF(IP.EQ.'G') RETURN
CALL LOOP(P1,P2,J,N
IF(IP.EQ.'0') GO TO 300
IF(IP.EQ.'W') GO TO 980
IF(IP.EQ.'T') GO TO 400
IF(IP.EQ.'A') GO TO 500
IF(IP.EQ.'O') GO TO 600
IF(IP.EQ.'c') GO TO 1600
IF(J.GT.N) GO TO 100
IF(IP.EQ.'!') MJ=J
IF(IP.EQ.'!') GO TO 2
IF(IP.EQ.' '.OR.IP.EQ.'?') GO TO 970
IF(IP.EQ.'F') IFIX(J)=7
IF(IP.EQ.'X') IFIX(J)=IFIX(J)+1
IF(IP.EQ.'Y') IFIX(J)=IFIX(J)+2
IF(IP.EQ.'M') IFIX(J)=IFIX(J)+4
IF(IP.EQ.'U') IFIX(J)=0
IF(IP.EQ.' ') GO TO 900
IF(IP.EQ.'L') GO TO 700
IF(IP.EQ.'U') GO TO 800
IF(IP.EQ.'U') IFIX(J)=8
IF(IP.EQ.'V') GO TO 800
XN2=1
UMAX=0.00001
XMLESS=99999999.
DO 35 I=1,N
```

```
IF(IFIX(I).EQ.7) GO TO 35
XN2=XN2+1
IF(XM(I).LT.XMLESS) XMLESS=XM(I)
IF(W(I).GT.WMAX) WMAX=W(I)
35 CONTINUE
SKN=SQRT(XN2)*WMAX
SKN=10.*SKN
DT=0.05*SQRT(XMLESS/SKN)
DT=2.*DT
GO TO 100
8 DO 7 I=1,N
X(I)=X(I)+XMJ
7 Y(I)=Y(I)+YMJ
XMJ=0.
YMJ=0.
IF(MJ.EQ.0) GO TO 5
XMJ=X(MJ)
YMJ=Y(MJ)
CALL UMOVE(X(MJ),Y(MJ))
ROT=-57.2957795*AMOD(T(MJ)-TIN(MJ),6.28318531)
CALL UROTAT(ROT)
XMIN=X(MJ)-500.
XMAX=XMIN+1000.
YMIN=Y(MJ)-375.
YMAX=YMIN+750.
CALL UWINDO(XMIN,XMAX,YMIN,YMAX)
5 CONTINUE
DO 8 I=1,N
X(I)=X(I)-XMJ
8 Y(I)=Y(I)-YMJ
IF(IP.NE.' ') IP='D'
IF(IP.EQ.'D'.OR.IP.EQ.'E') GO TO 3
CALL VECTOR(N,JP,NFIRST,MJ)
GO TO 100
3 CALL USET('INTEGER')
CALL UOUTLN
DO 4 I=1,N
CALL CIRCLE(X(I),Y(I),R(I),S(I),T(I))
XI=I
CALL UPRINT(X(I)+3.,Y(I)+3.,XI)
4 CONTINUE
GO TO 100
11 CONTINUE
XO=200.
YO=620.
CALL UMOVE(XO,YO+10.)
CALL UPEN(XO,YO+20.)
CALL UCRCLE(XO,YO,25.)
DPHE=0.62831853
PHE=0.0
DO 10 I=1,10
XD=25.*SIN(PHE)+XO
YD=25.*COS(PHE)+YO
CALL UMOVE(XD,YD)
XD=30.*SIN(PHE)+XO
YD=30.*COS(PHE)+YO
CALL UPEN(XD,YD)
XD=45.*SIN(PHE)+XO
YD=45.*COS(PHE)+YO
CALL USET('INTE')
XI=I-1
CALL UPRINT(XD,YD,XI)
10 PHE=PHE+DPHE
PHE=TIME*DPHE
CALL UMOVE(XO,YO)
XD=25.*SIN(PHE)+XO
YD=25.*COS(PHE)+YO
ITIME=TIME/10.
TIM=ITIME
CALL UPEN(XD,YD)
CALL UPRINT(XO,YO+60.,TIM)
CALL UMOVE(XO,YO+15.)
ANG=AMOD(PHE,6.2838153)*(-57.296)
CALL UARC(XO,YO,ANG)
TIM=NUE
CALL SCREEN(XP,YP)
CALL UPRINT(XP,YP,TIM)
GO TO 100
C     SET VELOCITIES TO ZERO
200 DO 210 I=1,N
VT(I)=0.0
VX(I)=0.0
210 VY(I)=0.0
GO TO 100
C     ASSIGN FRICTION VALUES
300 CALL USET('REAL')
CALL SCREEN(XP,YP)
CALL UREAD(XP,YP,F,1.,FL)
IF(ABS(F).LT.1.0) GO TO 303
F=SIN(F/57.296)/COS(F/57.296)
303 CONTINUE
IF(J.GT.N) GO TO 310
FRC(J)=F
GO TO 100
310 DO 320 I=1,N
320 FRC(I)=F
GO TO 100
C     FIX ALL ELEMENTS
350 DO 351 I=1,N
351 IFIX(I)=7
GO TO 100
C     ASSIGN  TENSION VALUES
400 CALL USET('REAL')
CALL SCREEN(XP,YP)
CALL UREAD(XP,YP,F,1.,FL)
IF(J.GT.N) GO TO 410
TEN(J)=F
GO TO 100
410 DO 420 I=1,N
420 TEN(I)=F
GO TO 100
C     COMPUTE DISTANCE AND ANGLE BETWEEN POINTS
500 CALL USET('USERAXIS')
CALL UGRIN(P3,P4,IP)
CALL USET('SYSTEMAXIS')
CALL LOOP(P3,P4,J,N)
VL(1)=SQRT((P1-P3)**2+(P2-P4)**2)
VL(2)=ATAN2((P4-P2),(P3-P1))*57.296
CALL SCREEN(XP,YP)
CALL USET('XYCOOR')
CALL UPRINT(XP,YP,VL)
GO TO 100
C     OUTPUT NUMERICAL RESULTS
600 CALL PLTFOR(XMIN,XMAX,YMIN,YMAX)
GO TO 100
C     INPUT LOADINGS
700 CONTINUE
CALL SCREEN(XP,YP)
```

```
CALL USET('REAL')
CALL UREAD(XP,YP,VEL,3.,FL)
FX(J)=VEL(1)
FY(J)=VEL(2)
FM(J)=2.*VEL(3)*(R(J)+S(J))
GO TO 100
C      INPUT VELOCITIES
800 CONTINUE
CALL SCREEN(XP,YP)
CALL USET('REAL')
CALL UREAD(XP,YP,VEL,3.,FL)
VX(J)=VEL(1)
VY(J)=VEL(2)
VT(J)=VEL(3)
GO TO 100
900 LENGTH=M*26
DO 950 K=J,N
DO 955 I=K,LENGTH,M
IF(I.GT.LENGTH) GO TO 960
955 IFIX(I)=IFIX(I+1)
960 CONTINUE
NAA(K)=NAA(K+1
NL=NAA(K)
957 IF(ABS(AA(NL)).LT.0.99) GO TO 950
AA(NL)=(ABS(AA(NL))-1.0)*SIGN(1.,AA(NL))
NL=NL+5
GO TO 957
950 CONTINUE
N=N-1
GO TO 100
970 CONTINUE
CALL UWINDO(0.,1000.,0.,750.)
CALL UGRIN(XP,YP,IP)
CALL USET('HARDWARE')
CALL USET('TEXT')
CALL UPRINT(XP,YP,' ')
DATA=J
CALL UPRNT1(DATA,'REAL')
CALL UPRNT1(FXSUM(J),'REAL')
CALL UPRNT1(FYSUM(J),'REAL')
CALL UPRNT1(FMSUM(J),'REAL')
CALL USET('TEXT')
CALL UPRINT(XP+10.,YP-20.,' ')
CALL UPRNT1(VX(J),'REAL')
CALL UPRNT1(VY(J),'REAL')
CALL UPRNT1(VT(J),'REAL')
CALL USET('TEXT')
CALL UPRINT(XP+10.,YP-40.,' ')
DATA=IFIX(J)
CALL UPRNT1(DATA,'INTE')
CALL UPRNT1(UGTF(J)*W(J),'REAL')
CALL UPRNT1(FRC(J),'REAL')
CALL UPRNT1(TEN(J),'REAL')
GO TO 100
980 CALL USET('REAL')
CALL SCREEN(XP,YP)
CALL UREAD(XP,YP,WF,1.,FL)
IF(J.GT.N) GO TO 985
UGTF(J)=WF
GO TO 100
985 DO 986 I=1,N
986 UGTF(I)=WF
GO TO 100
1200 CALL USET('USERAXIS')
CALL UGRIN(P11,P22,IP)
XMIN=AMIN1(P1,P11)
XMAX=AMAX1(P1,P11)
YMIN=AMIN1(P2,P22)
YMAX=AMAX1(P2,P22)
YMAX=YMIN+0.75*(XMAX-XMIN)
CALL UWINDO(XMIN,XMAX,YMIN,YMAX)
GO TO 2
1300 XMIN=0.0
XMAX=1000.
YMIN=0.0
YMAX=750.
CALL UWINDO(0.,1000.,0.,750.)
MJ=0
DO 6 I=1,N
X(I)=X(I)+XMJ
6 Y(I)=Y(I)+YMJ
CALL USET('SYSTEMAXIS')
1350 CONTINUE
CALL UDAREA(0.,1000.,0.,750.)
XMN=0.
XMX=1000.
YMN=0.
YMX=750.
XMJ=0.
YMJ=0.
CALL UMOVE(0.,0.)
CALL UROTAT(0.)
CALL USET('SYSTEMAXIS')
IF(IP.EQ.'N') CALL GIDVUP(NFIRST)
GO TO 2
1400 CALL USET('USERAXIS')
CALL UGRIN(P11,P22,IP)
XMN=AMIN1(P1,P11)
XMX=AMAX1(P1,P11)
YMN=AMIN1(P2,P22)
YMX=AMAX1(P2,P22)
YMX=YMN+0.75*(XMX-XMN)
CALL UDAREA(XMN,XMX,YMN,YMX)
GO TO 2
1600 CONTINUE
CALL SCREEN(XP,YP)
CALL USET('XYCOOR')
VL(1)=P1
VL(2)=P2
CALL UPRINT(XP,YP,VL)
GO TO 100
1700 CALL SCREEN(XP,YP)
CALL USET('REAL')
CALL UREAD(XP,YP,XMJ,1.,FL)
MJ=XMJ
IF(MJ.GT.N) MJ=0
GO TO 2
1800 CALL USET('XYCOOR')
CALL SCREEN(XP,YP)
VL(1)=SKN
VL(2)=DT
CALL UPRINT(XP,YP,VL)
GO TO 100
END
SUBROUTINE LOOP(P1,P2,K,N)
PARAMETER M=50
COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
COMMON/PARA/IFIX(M),UGTF(M),FRC(M),TEN(M)
```

```
DO 100 I=1,N
IF(((P1-X(I))**2+(P2-Y(I))**2).GT.50.) GO TO 100
P1=X(I)
P2=Y(I)
K=I
RETURN
100 CONTINUE
K=N+1
RETURN
END
SUBROUTINE VECTOR(N,JP,NFIRST,MJ)
PARAMETER M=50
DIMENSION XL(M),YL(M),TL(M)
COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
COMMON/PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
DO 30 I=1,N
IF(NFIRST.EQ.2) GO TO 50
R1=R(I)+S(I)
X1=X(I)+R1
X2=X(I)-R1
Y1=Y(I)+R1
Y2=Y(I)-R1
21 CONTINUE
IF(JP.GT.0) GO TO 25
CALL UMOVE(XL(I),YL(I))
CALL UPEN(X(I),Y(I))
GO TO 50
25 CONTINUE
IF(MJ.NE.0) GO TO 40
IF(IFIX(I).EQ.7) GO TO 30
IF(ABS((T(I)-TL(I))*(R(I)+S(I))).GT.1.) GO TO 40
IF(ABS(X(I)-XL(I)).GT.1.) GO TO 40
IF(ABS(Y(I)-YL(I)).GT.1.) GO TO 40
GO TO 30
40 CALL CIRCLE(X(I),Y(I),R(I),S(I),T(I))
50 XL(I)=X(I)
YL(I)=Y(I)
TL(I)=T(I)
30 CONTINUE
RETURN
END
SUBROUTINE SCREEN (XP,YP)
DATA X/800./,Y/740./
CALL UWINDO(0.,1000.,0.,750.)
Y=Y-20.
IF(Y.GT.100.) GO TO 10
Y=740.
10 CONTINUE
XP=X
YP=Y
RETURN
END
SUBROUTINE GEN(N,R,X1,Y1)
PARAMETER M=50
COMMON/COOR/X(M),Y(M),T(M),RR(M),S(M),W(M),XM(M)
DIMENSION D(3)
D(3)=0.0
CALL USET('REALNUMBER')
CALL SCREEN(XP,YP)
CALL UREAD(XP,YP,D,3.,FL)
MM=N
II=D(1)
JJ=D(2)
KK=ABS(D(3))+.1
IF(D(3).LT.0.0) KK=-KK
Q=0.0
DO 100 J=1,JJ
DO 200 I=1,II
Z=I-1
X(N)=X1+Z*2.0*R+Q
Y(N)=Y1
RR(N)=R
200 N=N+1
II=II+KK
B=J/2
C=FLOAT(J)/2.0
Q=2.0*(B-C)*R*SIGN(1.0,D(3))-(2.0*B*R*D(3))
100 Y1=SQRT(3.0)*R+Y1
DO 300 I=MM,N-1
300 CALL UCRCLE(X(I),Y(I),R)
CALL USET('TEXT')
RETURN
END
SUBROUTINE WEIGHT(WF)
CALL SCREEN(XP,YP)
CALL USET('REAL')
CALL UREAD(XP,YP,WF,1.,FL)
RETURN
END
SUBROUTINE DAMP(DF)
CALL SCREEN(XP,YP)
CALL USET('REAL')
CALL UREAD(XP,YP,DF,1.,FL)
RETURN
END
SUBROUTINE INTERVAL(KASK)
CALL SCREEN(XP,YP)
CALL USET('REAL')
CALL UREAD(XP,YP,X,1.,FL)
KASK=X+.1
RETURN
END
SUBROUTINE CIRCLE(X,Y,R,S,T)
IF(S.LT.0.01) GO TO 100
CALL UMOVE(X-5.*COS(T),Y-5.*SIN(T))
CALL UPEN(X+5.*COS(T),Y+5.*SIN(T))
CALL UMOVE(X-5.*SIN(T),Y+5.*COS(T))
CALL UPEN(X+5.*SIN(T),Y-5.*COS(T))
X1=X+R*SIN(T)+S*COS(T)
Y1=Y-R*COS(T)+S*SIN(T)
XC1=X+S*COS(T)
YC1=Y+S*SIN(T)
X2=X-R*SIN(T)-S*COS(T)
Y2=Y+R*COS(T)-S*SIN(T)
XC2=X-S*COS(T)
YC2=Y-S*SIN(T)
CALL UMOVE(X1,Y1)
CALL UARC(XC1,YC1,180.)
CALL UPEN(X2,Y2)
CALL UARC(XC2,YC2,180.)
CALL UPEN(X1,Y1)
RETURN
100 CALL UCRCLE(X,Y,R)
CALL UMOVE(X,Y)
TEM=AMOD(T,6.2831853)
XC=X+R*COS(TEM)
YC=Y+R*SIN(TEM)
CALL UPEN(XC,YC)
```

```
RETURN
END
SUBROUTINE MOTION(NWW)
PARAMETER M=50
PARAMETER MSZ=M*24
PARAMETER MLIST=M*6
PARAMETER LBACK=MSZ-5
COMMON/NUMS/NWO,NW3
COMMON/JUNK/N,SKN,DKN,TIME,DT,WF,DF
COMMON/LIST/LL(MLIST)
COMMON/STORE/AA(MSZ),NA(M),NAA(M)
COMMON/FORCE/FXSUM(M),FYSUM(M),FMSUM(M),FX(M),FY(M),FM(M)
COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
COMMON/VELO/UX(M),UY(M),UT(M),VX(M),VY(M),VT(M)
COMMON/INIT/XIN(M),YIN(M),TIN(M)
COMMON/PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
DATA PI/3.1415926/
IF(NW3.EQ.0) NW=N+1
INT=NWW
KOUNT=0
ICYCLE=1
DO 5000 ICY=1,50
KOUNT=KOUNT+1
IF(KOUNT.EQ.INT) ICYCLE=1
IF(KOUNT.EQ.INT) KOUNT=0
TIME=TIME+DT
DO 90 I=1,N
WF=WGTF(I)
FXSUM(I)=FX(I)
FYSUM(I)=FY(I)-W(I)*WF
90 FMSUM(I)=FM(I)
NL=1
I=1
IF(ICYCLE.NE.1) GO TO 700
NWW=0
NWO=NW
NWD=NW
I=1
KK=1
C     **********
700 CONTINUE
IF(ICYCLE.EQ.1) GO TO 705
NLL=NWO
NRET=1
J=ABS(AA(NLL))
IF(J.EQ.0) NLL=NLL+1
IF(J.EQ.0) GO TO 203
ILOC1=NLL+2
ILOC2=NLL+3
GO TO 706
705 CONTINUE
J=LL(KK)
IF(J.LT.0) GO TO 100
KK=KK+1
IF(J.EQ.I) GO TO 202
IF(IFIX(I).GE.7.AND.IFIX(J).GE.7) GO TO 700
NRET=1
D=SQRT((X(I)-X(J))**2+(Y(I)-Y(J))**2)
IF(D.GT.(R(I)+R(J)+S(I)+S(J)+2.)) GO TO 200
706 NAX=1
IF(S(I).LT.0.1) NAX=4
IF(S(J).LT.0.1) NAX=3
IF(S(I).LT.0.1.AND.S(J).LT.0.1) NAX=2
C     **********
GO TO (1,2,3,4),NAX
2 D=SQRT((X(J)-X(I))**2+(Y(I)-Y(J))**2)-2.
LOC=1
RQ=R(I)+R(J)
IF(D.GT.RQ) GO TO 200
B=ATAN2(Y(J)-Y(I),X(J)-X(I))
XC=X(I)+R(I)*COS(B)
YC=Y(I)+R(I)*SIN(B)
B=B-PI/2.
NRET=1
C     **********
14 IF(ICYCLE.NE.1) GO TO 150
NL=NAA(I)
XJ=J
XLOC=LOC
XJ=XJ+ABS(XC)/10000.
IF(XC.LT.0.0) XJ=-XJ
XLOC=XLOC+ABS(YC)/10000.
IF(YC.LT.0.0) XLOC=-XLOC
15 NUM=ABS(AA(NL))
NCOR=ABS(AA(NL+1))
NQ=NL
NL=NL+5
IF(NL.GT.LBACK) NL=1
NAA(I)=NL
IF(NUM.EQ.0) GO TO 20
IF(NUM.GT.J) GO TO 20
IF(NUM.EQ.J.AND.NCOR.EQ.LOC) GO TO 25
GO TO 15
20 D=D+1.9
IF(TEN(I).GT.0.0.OR.TEN(J).GT.0.0) D=D-1.
IF(D.GT.RQ) GO TO 200
AA(NW)=XJ
AA(NW+1)=XLOC
AA(NW+2)=0.0
AA(NW+3)=0.0
AA(NW+4)=B
GO TO 23
25 AA(NW)=XJ
AA(NW+1)=XLOC
AA(NW+2)=AA(NQ+2)
AA(NW+3)=AA(NQ+3)
AA(NW+4)=B
23 ILOC1=NW+2
ILOC2=NW+3
NW=NW+5
IF(NW.GT.LBACK) NW=1
NWW=NWW+1
GO TO 150
C     **********
3 IK=I
JK=J
K=1
LOC=1
NRET=1
GO TO 160
4 IK=J
JK=I
K=1
NRET=1
LOC=1
GO TO 160
1 IK=I
JK=J
```

```
K=1
NRET=2
LOC=1
GO TO 160
11 IK=I
JK=J
K=2
NRET=3
LOC=2
GO TO 160
21 IK=J
JK=I
K=1
NRET=4
LOC=3
GO TO 160
31 IK=J
JK=I
K=2
NRET=1
LOC=4
C     **********
160 RQ=R(IK)+R(JK)
B=T(IK)
CO=COS(B)
SI=SIN(B)
Z=S(IK)
X1=X(IK)-Z*CO
X2=X(IK)+Z*CO
Y1=Y(IK)-Z*SI
Y2=Y(IK)+Z*SI
SIG=(-1)**K
P1=X(JK)+SIG*COS(T(JK))*S(JK)
P2=Y(JK)+SIG*SIN(T(JK))*S(JK)
CO=COS(-B)
SI=SIN(-B)
XN=(P1-X1)*CO-(P2-Y1)*SI
YN=(P1-X1)*SI+(P2-Y1)*CO
D=ABS(YN)-2.0
IF(ABS(YN).GT.(RQ+2.)) GO TO 200
IF(XN.LT.(-RQ-2.0)) GO TO 200
IF(XN.GT.(2.0+RQ+2.*S(IK))) GO TO 200
IF(XN.GT.0.0.AND.XN.LT.(2.0*S(IK))) GO TO 155
IF(XN.GT.0.0) GO TO 56
D=SQRT((X1-P1)**2+(Y1-P2)**2)-2.0
IF(D.GT.RQ) GO TO 200
B=ATAN2(P2-Y1,P1-X1)
XC=X1+R(IK)*COS(B)
YC=Y1+R(IK)*SIN(B)
B=B-PI/2.
IF(I.NE.IK) B=B-PI
GO TO 166
56 D=SQRT((X2-P1)**2+(Y2-P2)**2)-2.0
IF(D.GT.RQ) GO TO 200
B=ATAN2(P2-Y2,P1-X2)
XC=X2+R(IK)*COS(B)
YC=Y2+R(IK)*SIN(B)
B=B-PI/2.
IF(I.NE.IK) B=B-PI
GO TO 166
155 CO=COS(B)
SI=SIN(B)
IF(YN.GT.0.0) YN=YN-R(JK)
IF(YN.LT.0.0) YN=YN+R(JK)
```

```
XC=XN*CO-YN*SI+X1
YC=XN*SI+YN*CO+Y1
IF(YN.GT.0.0) B=B-PI
IF(I.EQ.IK) B=B-PI
166 GO TO 14
C     **********
150 CONTINUE
DY=UY(J)-UY(I)+(UT(J)*(XC-X(J)))
&    -(UT(I)*(XC-X(I)))
DX=UX(J)-UX(I)-(UT(J)*(YC-Y(J)))
&    +(UT(I)*(YC-Y(I)))
FRIC=ABS(FRC(I))
IF(FRC(J).LT.FRC(I)) FRIC=ABS(FRC(J))
DUS=DX*COS(B)+DY*SIN(B)
DUN=DY*COS(B)-DX*SIN(B)
AA(ILOC1)=AA(ILOC1)-DUN*SKN
AA(ILOC2)=AA(ILOC2)+DUS*SKN
XMA=SQRT(XM(I)*XM(J))
DKN=1.95*SQRT(SKN*XMA)/DT
DN=-DUN*DKN*DF
DS=DUS*DKN*DF
TN=ABS(TEN(I))
IF(TEN(J).LT.TEN(I)) TN=ABS(TEN(J))
IF(AA(ILOC1).GT.(-TN)) GO TO 1000
AA(ILOC1)=0.0
AA(ILOC2)=0.0
DN=0.0
DS=0.0
GO TO 1500
1000 IF(ABS(AA(ILOC2)).LT.(FRIC*ABS(AA(ILOC1)))) GO TO 1500
DS=0.0
AA(ILOC2)=FRIC*ABS(AA(ILOC1))*SIGN(1.,AA(ILOC2))
1500 AN=AA(ILOC1)
AS=AA(ILOC2)
FYI=(AS+DS)*SIN(B)-(AN+DN)*COS(B)
FXI=(AS+DS)*COS(B)+(AN+DN)*SIN(B)
FMI=FYI*(XC-X(I))-FXI*(YC-Y(I))
FYJ=-FYI
FXJ=-FXI
FMJ=FYJ*(XC-X(J))-FXJ*(YC-Y(J))
FXSUM(I)=FXSUM(I)+FXI
FXSUM(J)=FXSUM(J)+FXJ
FYSUM(I)=FYSUM(I)+FYI
FYSUM(J)=FYSUM(J)+FYJ
FMSUM(I)=FMSUM(I)+FMI
FMSUM(J)=FMSUM(J)+FMJ
NLL=NLL+5
ILOC1=NLL+2
ILOC2=NLL+3
200 CONTINUE
C     **********
GO TO (201,11,21,31),NRET
201 GO TO 700
202 CONTINUE
AA(NU)=0.0
IF(I.LE.N) NU=NU+1
IF(NU.GT.LBACK) NU=1
IF(I.LE.N) NAA(I)=NUD
NUD=NU
203 CONTINUE
I=I+1
IF(I.GT.N) GO TO 100
GO TO 700
100 CONTINUE
```

```
      DO 105 I=1,N
      NF=IFIX(I)
      IF(NF.EQ.8) GO TO 75
      UX(I)=UX(I)+FXSUM(I)*DT/XM(I)
      UY(I)=UY(I)+FYSUM(I)*DT/XM(I)
      UT(I)=UT(I)+FMSUM(I)*DT/(XM(I)*((R(I)+S(I))**2))
      IF(NF.EQ.0) GO TO 75
      IF(NF.NE.7) GO TO 50
      UX(I)=0.0
      UY(I)=0.0
      UT(I)=0.0
      GO TO 75
50    IF(NF.NE.1.AND.NF.NE.3.AND.NF.NE.5)
     & GO TO 55
      UX(I)=0.
55    IF(NF.NE.2.AND.NF.NE.3.AND.NF.NE.6)
     &  GO TO 60
      UY(I)=0.
60    IF(NF.NE.4.AND.NF.NE.5.AND.NF.NE.6)
     & GO TO 75
      UT(I)=0.
75    UX(I)=UX(I)*DT
      UY(I)=UY(I)*DT
      UT(I)=UT(I)*DT
      X(I)=X(I)+UX(I)
      Y(I)=Y(I)+UY(I)
      T(I)=T(I)+UT(I)
105   CONTINUE
      ICYCLE=0
5000  CONTINUE
      NW3=NW
      RETURN
      END
      SUBROUTINE PLIST(K)
      PARAMETER M=50
      PARAMETER MLIST=M*6
      COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
      COMMON/VELO/UX(M),UY(M),UT(M),UX(M),UY(M),UT(M)
      COMMON/JUNK/N,SKN,DKN,TIME,DT,WF,DF
      COMMON/PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
      COMMON/LIST/LL(MLIST)
      RNG=5.
      K=1
      DO 100 I=1,N
      DO 200 J=I,N
      IF(I.EQ.J) GO TO 200
      D=(X(I)-X(J))**2+(Y(I)-Y(J))**2
      RG=(R(I)+R(J)+S(I)+S(J)+RNG)**2
      IF(D.GT.(RG+100.)) GO TO 200
      IF(D.GT.RG) GO TO 190
      LL(K)=J
      K=K+1
      GO TO 200
190   D0=SQRT(D)
      WX=50.*DT*(UX(J)-UX(I))
      WY=50.*DT*(UY(J)-UY(I))
      DN=SQRT((X(J)-X(I)+WX)**2+(Y(J)-Y(I)+WY)**2)
      RNG=D0-DN
      IF(RNG.LT.0.0) GO TO 200
      RG=R(I)+R(J)+S(I)+S(J)+RNG
      IF(D0.GT.RG) GO TO 200
      LL(K)=J
      K=K+1
200   CONTINUE
      LL(K)=I
      K=K+1
100   CONTINUE
      LL(K)=-1
      RETURN
      END
      SUBROUTINE PLTFOR(XMIN,XMAX,YMIN,YMAX)
      PARAMETER M=50
      PARAMETER MSZ=M*24
      PARAMETER LBACK=MSZ-5
      COMMON/NUMS/NWO,NWL
      COMMON/JUNK/N
      COMMON/STORE/AA(MSZ),NA(M),NAA(M)
      COMMON/PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
      CALL WWINDO(0.,1000.,0.,750.)
      CALL USET('HARDWARE')
      CHARACTER IP*1
      CALL WGRIN(XP,YP,IP)
      XN=N
      AN=0.000001
      I=NWO
      XI=1.
1     IF(I.GT.LBACK) I=1
      IF(XI.GT.XN) GO TO 10
      IF(ABS(AA(I)).GT.1.0) GO TO 5
      I=I+1
      XI=XI+1.
      GO TO 1
5     IF(ABS(AA(I+2)).GT.AN) AN=ABS(AA(I+2))
      IF(ABS(AA(I+3)).GT.AN) AN=ABS(AA(I+3))
      I=I+5
      GO TO 1
10    SC=20./AN
      XI=1.
      I=NWO
2     IF(I.GT.LBACK) I=1
      IF(XI.GT.XN) RETURN
      IF(ABS(AA(I)).GT.1.0) GO TO 15
      I=I+1
      XI=XI+1.
      GO TO 2
15    CONTINUE
      NC=AA(I)
      XJ=IABS(NC)
      XC=NC
      XC=(AA(I)-XC)*10000.
      NC=AA(I+1)
      YC=NC
      YC=(AA(I+1)-YC)*10000.
      B=AA(I+4)
      AN=SC*ABS(AA(I+2))
      AS=SC*ABS(AA(I+3))
      IF(IP.NE.'H') GO TO 3
      CALL USET('REAL')
      CALL WWINDO(0.,1000.,0.,750.)
      CALL USET('SYSTEMAXIS')
      YP=YP-20.
      IF(YP.LT.50.) XP=XP-250.
      IF(YP.LT.50.) YP=700.
      II=XI
      JJ=XJ
      F=AMIN1(ABS(FRC(II)),ABS(FRC(JJ)))
      F=F*AA(I+2)
      IF(ABS(F).LT.0.00001) GO TO 17
```

```
F=ABS(AA(I+3)/F)
17 CONTINUE
CALL UPRINT(XP,YP,AA(I+3))
CALL UPRINT(XP-75.,YP,AA(I+2))
CALL USET('INTEGER')
CALL UPRINT(XP-115.,YP,XJ)
CALL UPRINT(XP-150.,YP,XI)
3 CONTINUE
CALL USET('USERAXIS')
CALL UWINDO(XMIN,XMAX,YMIN,YMAX)
XP1=XC+COS(B)*AS
XP2=XC-COS(B)*AS
YP1=YC+SIN(B)*AS
YP2=YC-SIN(B)*AS
CALL UMOVE(XP1,YP1)
CALL UPEN(XP2,YP2)
B=B-1.5707963
XP1=XC+COS(B)*AN
XP2=XC-COS(B)*AN
YP1=YC+SIN(B)*AN
YP2=YC-SIN(B)*AN
CALL UMOVE(XP1,YP1)
CALL UPEN(XP2,YP2)
I=I+5
GO TO 2
END
SUBROUTINE SAVE(N,J)
PARAMETER M=50
PARAMETER MM=5*M
PARAMETER MMM=4*M
COMMON/COOR/X(M),Y(M),T(M),R(M),S(M),W(M),XM(M)
COMMON /PARA/IFIX(M),WGTF(M),FRC(M),TEN(M)
DIMENSION DATA(MM),DAT(MMM)
IF(J.EQ.1) GO TO 10
DO 5 I=1,MM
5 X(I)=DATA(I)
DO 6 I=1,MMM
6 IFIX(I)=DAT(I)
N=NN+1
RETURN
10 DO 20 I=1,MM
20 DATA(I)=X(I)
DO 25 I=1,MMM
25 DAT(I)=IFIX(I)
NN=N
RETURN
END
SUBROUTINE GRID(XMIN,XMAX,YMIN,YMAX)
CALL USET('USERAXIS')
10 X=-1000.
20 CALL UMOVE(X,-1000.)
CALL UPEN(X,1000.)
X=X+50.
IF(X.GT.1000.) GO TO 30
GO TO 20
30 Y=-1000.
40 CALL UMOVE(-1000.,Y)
CALL UPEN(1000.,Y)
Y=Y+50.
IF(Y.GT.1000.) RETURN
GO TO 40
END
SUBROUTINE UPAUSE
CALL UWAIT(3.)
RETURN
END
```

In accordance with letter from DAEN-RDC, DAEN-ASI dated 22 July 1977, Subject: Facsimile Catalog Cards for Laboratory Technical Publications, a facsimile catalog card in Library of Congress MARC format is reproduced below.

[illegible]
Analysis [illegible] assemblages of [illegible] interactive graphics [illegible] by [illegible] B. Palmerton [illegible]
U.S. Army Engineer Waterways Experiment Station [illegible]
assistant Secretary of the Army [illegible]
Washington, D. C., under [illegible]
Unit [illegible] -- Vicksburg, Miss. : [illegible] Army Engineer Waterways
Experiment Station ; Springfield, Va. : available from [illegible],
[illegible].
77, [illegible] p. : ill. ; [illegible] cm. -- (Miscellaneous paper - U.S.
Army Engineer Waterways Experiment Station ; [illegible])
Cover title.
"December [illegible]."
References: p. 77.
[illegible]
[illegible]
Development). [illegible] United States. Army Engineer Waterways
Experiment Station. [illegible]
[illegible]
Waterways Experiment Station) ; [illegible]
TA7.W34m [illegible]

DAT
FILM
3